Received: 31 July 2017 | Revised: 13 December 2019 | Accepted: 3 April 2021

DOI: 10.1002/jgt.22683

ARTICLE

WILEY

Recognizing graphs close to bipartite graphs with an application to colouring reconfiguration

Marthe Bonamy[1] | **Konrad K. Dabrowski**[2] | **Carl Feghali**[3] | **Matthew Johnson**[4] | **Daniël Paulusma**[4]

[1]CNRS, LaBRI, Université de Bordeaux, Bordeaux, France

[2]School of Computing, University of Leeds, Leeds, UK

[3]Computer Science Institute, Charles University, Prague, Czech Republic

[4]Department of Computer Science, Durham University, Durham, UK

Correspondence
Matthew Johnson, Department of Computer Science, Durham University, Mathematical Sciences & Computer Science Building, Upper Mountjoy Campus, Stockton Road, Durham DH1 3LE, UK
Email: matthew.johnson2@durham.ac.uk

Funding information

Leverhulme Trust, Grant/Award Number: RPG-2016-258; Fondation Sciences Mathematiques de Paris; London Mathematical Society, Grant/Award Number: 41536; Grantová Agentura Ceské Republiky, Grant/Award Number: 19-21082S; Engineering and Physical Sciences Research Council, Grant/Award Number: EP/K025090/1

Abstract

We continue research into a well-studied family of problems that ask whether the vertices of a given graph can be partitioned into sets A and B, where A is an independent set and B induces a graph from some specified graph class $\mathcal{G}$. We consider the case where $\mathcal{G}$ is the class of k-degenerate graphs. This problem is known to be polynomial-time solvable if $k = 0$ (recognition of bipartite graphs), but NP-complete if $k = 1$ (near-bipartite graphs) even for graphs of maximum degree 4. Yang and Yuan showed that the $k = 1$ case is polynomial-time solvable for graphs of maximum degree 3. This also follows from a result of Catlin and Lai. We study the general $k \geq 1$ case for n-vertex graphs of maximum degree $k + 2$. We show how to find A and B in $O(n)$ time for $k = 1$, and in $O(n^2)$ time for $k \geq 2$. Together, these results provide an algorithmic version of a result of Catlin and also provide an algorithmic version of a generalization of Brook's Theorem, proved by Borodin et al. and Matamala. The results also enable us to solve an open problem of Feghali et al. For a given graph G and positive integer ℓ, the

vertex colouring reconfiguration graph of G has as its vertex set the set of ℓ-colourings of G and contains an edge between each pair of colourings that differ on exactly on vertex. We complete the complexity classification of the problem of finding a path in the reconfiguration graph between two given ℓ-colourings of a given graph of maximum degree k.

KEYWORDS
degenerate graphs, near-bipartite graphs, reconfiguration graphs

1 | INTRODUCTION

The COLOURING problem asks if a given graph is *k-colourable* for some given integer k; that is, if the vertices of the graph can be coloured with at most k colours, such that no two adjacent vertices are coloured alike. This is a central problem in graph theory and well known to be NP-complete even if $k = 3$ [35]. A stronger property of a graph is that of being $(k-1)$-*degenerate*, which is the case when every induced subgraph has a vertex of degree at most $k-1$. Every $(k-1)$-degenerate graph is k-colourable, but the converse is not true.

For an arbitrarily large integer k, there exist k-degenerate graphs that are not $(k-1)$-degenerate but that can be decomposed into a p-degenerate induced subgraph and a q-degenerate induced subgraph for two small integers p and q. For example, if we take complete bipartite graphs of degree k, then we can let $p = q = 0$. If a k-degenerate graph is decomposable in this way, it is not only $(k+1)$-colourable but even $(p+q+2)$-colourable. This leads to the well-studied problem of identifying graphs whose vertex sets can be partitioned into two sets A and B such that A and B induce a p-degenerate graph and a q-degenerate graph, respectively; see, for instance, [9,10,32,39,45,46]. If a graph has such a partition with $p + q = \ell - 2$, then we can say that the graph is "robustly" ℓ-colourable. For the sake of example: every planar graph is 5-degenerate, which implies that it is 6-colourable, but we can improve this to 5-colourable by applying the result of Thomassen [46], which states that every planar graph can be decomposed into a 0-degenerate graph and a 3-degenerate graph, or by applying another result of Thomassen [45], which states that every planar graph can be decomposed into a 1-degenerate graph and a 2-degenerate graph. So being 5-colourable is a "robust" property of planar graphs (in contrast to being 4-colourable; there is no p and q with $p + q \leq 2$ such that we can guarantee a decomposition of a planar graph into a p-degenerate graph and a q-degenerate graph; see [29] for $p = 0, q = 2$ and [20] for $p = q = 1$).

Research Question. We will apply the notion of "robust" k-colourability to a central problem in the area of graph reconfiguration. The k-colouring reconfiguration graph $R_k(G)$ has as vertices the k-colourings of G and two k-colourings are adjacent if and only if they differ on exactly one vertex of G. The problem of finding a path between two given k-colourings in $R_k(G)$ is PSPACE-hard even if $k = 4$ and G is planar bipartite [7]. In its complexity classification for graphs of maximum degree Δ [24] there is one open case (k, Δ) left. As argued in [24], to solve this case we must answer the following question:

Is it possible to find in polynomial time a partition (A, B) of the vertex set of a graph G of maximum degree k, such that A is 0-degenerate and B induces a $(k-2)$-degenerate graph?

1.1 | Known existence results

We note that in the above question we must *find* a partition. This is a different question than deciding whether such a partition *exists*. For the latter question, a number of results exist in the literature. We survey these results, as they are very insightful for our question, although they do not solve it.

We first note that if we ask whether a graph has a partition into a 0-degenerate graph and a q-degenerate graph, then we can see q as being a distance measure to control how "far" the graph is from being bipartite. As every 0-degenerate graph is an independent set, checking if the distance is 0 is the same as checking bipartiteness, which can be solved in linear time. Graphs within distance 1 from being bipartite are said to be *near-bipartite*. Such a graph has a *near-bipartite decomposition*, that is, a partition (A, B) of its vertex set, where A is an independent set and B induces a 1-degenerate graph, or equivalently, a forest. Deciding whether a graph is near-bipartite is NP-complete [14].

Yang and Yuan [49] proved that the problem remains NP-complete even for graphs of maximum degree 4, but becomes polynomial-time solvable for graphs of maximum degree 3. To prove the latter result, they showed that every connected graph of maximum degree at most 3 is near-bipartite except K_4 (we let K_k denote the complete graph on k vertices). This characterization was also shown by Catlin and Lai, who proved that the independent set A may even be assumed to be of maximum size.

Theorem 1 (Catlin and Lai [18]). *The vertex set of every connected graph of maximum degree 3 that is not isomorphic to K_4 can be partitioned into two sets A and B, where A is a maximum size independent set and B is a forest.*

Theorem 1 generalizes the $k = 3$ case of the following earlier result of Catlin.

Theorem 2 (Catlin [17]). *For every integer $k \geq 3$, the vertex set of every connected graph of maximum degree k that is not isomorphic to K_{k+1} can be partitioned into two sets A and B, where A is a maximum size independent set and B induces a graph that does not contain a K_k.*

Matamala generalized both Theorems 1 and 2.

Theorem 3 (Matamala [39]). *For every three integers $k \geq 3$ and $p, q \geq 0$ with $p + q = k - 2$, the vertex set of every connected graph of maximum degree k that is not isomorphic to K_{k+1} can be partitioned into two sets A and B, where A induces a p-degenerate subgraph of maximum size and B induces a q-degenerate subgraph.*

Before Theorem 3 appeared, Borodin, Kostochka and Toft proved a more general result, except that the property of the first set having maximum size is not included. We present a simpler version of their result and refer to [11] for full details.

Theorem 4 (Borodin et al. [11]). *For all integers $k \geq 3$, $s \geq 2$, and $p_1, ..., p_s \geq 0$ such that $p_1 + \cdots + p_s = k - s$, the vertex set of every connected graph of maximum degree k that is not isomorphic to K_{k+1} can be partitioned into sets $A_1, ..., A_s$, where A_i induces a p_i-degenerate subgraph for $i \in \{1, ..., s\}$.*

Brooks' Theorem [15] states that every graph G with maximum degree $k \geq 2$ is k-colourable unless G is a complete graph or an odd cycle. Recall that every $(k-1)$-degenerate graph is k-colourable. Hence Theorems 3 and 4, together with the trivial case $k = 2$, generalize Brooks' Theorem.

By choosing $p = 0$ in Theorem 3 we obtain the following special case, which implies that every connected graph of maximum degree k except K_{k+1} is within distance $k - 2$ of being bipartite.

Theorem 5. *For every integer $k \geq 3$, the vertex set of every connected graph of maximum degree k that is not isomorphic to K_{k+1} can be partitioned into two sets A and B, where A is a maximum size independent set and B induces a $(k-2)$-degenerate subgraph.*

Theorem 5 only guarantees that the existence of a partition (A, B) can be tested in polynomial time. It does not tell us how to find such a partition. Obtaining an algorithmic version of Theorem 5 corresponds to our research question. We cannot hope to keep the condition that A is a maximum size independent set, as this would require solving the NP-complete problem INDEPENDENT SET for connected cubic graphs [26]. Before giving our results, we first survey some other algorithmic results.

1.2 | Known algorithmic results

1.2.1 | Special graph classes

As discussed, Yang and Yuan [49] proved that recognizing near-bipartite graphs is polynomial-time solvable for graphs of maximum degree k when $k \leq 3$ and NP-complete when $k \geq 4$. They also proved that recognizing near-bipartite graphs of diameter k is polynomial-time solvable when $k \leq 2$ and NP-complete when $k \geq 4$. Recently, we solved their missing case by proving that recognizing near-bipartite graphs of diameter 3 is NP-complete [4]. Brandstädt et al. [12] proved that recognizing near-bipartite perfect graphs is NP-complete. We also proved that recognizing near-bipartite graphs is NP-complete for line graphs of maximum degree 4 [5].

Borodin and Glebov [10] showed that every planar graph of girth at least 5 is near-bipartite (see [32] for an extension of this result). Dross et al. [22] asked whether every triangle-free planar graph is near-bipartite. In fact, they proved that if this is not the case, then the problem of recognizing near-bipartite graphs is NP-complete for triangle-free planar graphs. They presented a construction that can be easily modified to prove that the problem of recognizing near-bipartite graphs is NP-complete for planar graphs.[1]

[1]The NP-hardness reduction in [22] uses a minimal triangle-free planar graph G that is not near-bipartite, and it is not known whether such graphs exist. If we remove the triangle-free condition, we can replace G by K_4.

1.2.2 | Minimum independent feedback vertex sets

The problem of finding a decomposition into an independent set A and a forest B where the size of A is minimum has also been studied. In this context A is said to be an *independent feedback vertex set*. Computing a minimum independent feedback vertex set has been shown to be NP-hard even for planar bipartite graphs of maximum degree 4, but linear-time solvable for graphs of bounded treewidth, chordal graphs and P_4-free graphs [44] (it was already known that near-bipartite P_4-free graphs can be recognized in linear time [12]). Recently, we extended the polynomial-time result from [44] for P_4-free graphs to P_5-free graphs [5]. We also gave a polynomial-time algorithm for computing a minimum independent feedback vertex set of a graph of diameter 2 [4].

The problem of computing small independent feedback vertex sets has also been studied from the perspective of parameterized complexity. In this setting, the size of A is taken as the parameter. Misra et al. [42] gave the first FPT algorithm for this problem, which was later improved by Agrawal et al. [1].

1.2.3 | Problem variants

The Induced Forest 2-Partition problem is closely related to the problem of recognizing near-bipartite graphs. It asks whether the vertex set of a given graph can be decomposed into two disjoint sets A and B, where both A and B induce forests. Wu, Yuan and Zhao [48] proved that Induced Forest 2-Partition is NP-complete for graphs of maximum degree 5 and polynomial-time solvable for graphs of maximum degree at most 4. The problem variant where the maximum degree of one of the two induced forests is bounded by some constant has also been studied, in particular from a structural point of view (see, for instance, [22]).

A similar problem, known as Dominating Induced Matching, asks whether the vertex set of a graph can be partitioned into an independent set and an induced matching (a set of isolated edges) and was shown by Grinstead et al. [27] to be NP-complete. Brandstädt et al. [14] proved NP-completeness of another closely related problem: deciding whether the vertex set of a given graph can be decomposed into an independent set and a tree. As trees, induced matchings and forests are 2-colourable, these two problems and that of recognizing near-bipartite graphs can be seen as restricted variants of the 3-colouring problem. This problem is well known to be NP-complete [36]. However, the NP-hardness result of Brandstädt et al. [12] for perfect graphs shows that there are hereditary graph classes on which the complexities of recognizing near-bipartite graphs and 3-colouring do not coincide, as 3-colouring (or even k-colouring with k part of the input [28]) is polynomial-time solvable for perfect graphs.

A 3-colouring of a graph is acyclic if *every* two colour classes induce a forest. We observe that every graph with an acyclic 3-colouring is near-bipartite, but the reverse is not necessarily true. Kostochka [33] proved that the corresponding decision problem Acyclic 3-colouring is NP-complete. Later, Ochem [43] showed that Acyclic 3-colouring is NP-complete even for planar bipartite graphs of maximum degree 4. As every bipartite graph is near-bipartite, this result implies that there are hereditary graph classes on which the complexities of recognizing near-bipartite graphs and Acyclic 3-colouring do not coincide.

1.2.4 | Generalizations

For a fixed graph class $\mathcal{G}$ (i.e. $\mathcal{G}$ is not part of the input), Brandstädt et al. [14] considered the following more general problem:

STABLE($\mathcal{G}$)
Instance: A graph $G = (V, E)$
Question: Can V be decomposed into two disjoint sets A and B, where A is an independent set and B induces a graph in $\mathcal{G}$?

Note that STABLE($\mathcal{G}$) is equivalent to 3-COLOURING if we choose $\mathcal{G}$ to be the class of bipartite graphs. If we choose $\mathcal{G}$ to be the class of $(k-2)$-degenerate graphs, then we obtain the decision version of the problem we consider in this article.

If $\mathcal{G}$ is the class of complete graphs, then STABLE($\mathcal{G}$) is the problem of recognizing split graphs, and this problem can be solved in polynomial time. Brandstädt et al. [14] proved that STABLE($\mathcal{G}$) is NP-complete when $\mathcal{G}$ is the class of trees or the class of trivially perfect graphs, and polynomial-time solvable when $\mathcal{G}$ is the class of co-bipartite graphs, the class of split graphs, or the class of threshold graphs. Moreover, STABLE($\mathcal{G}$) has also been shown to be NP-complete when $\mathcal{G}$ is the class of triangle-free graphs [16], the class of P_4-free graphs [30], the class of graphs of maximum degree 1 [38], or, more generally, a class of graphs that has any additive hereditary property not equal to or divisible by the property of being edgeless [34], whereas it is also polynomial-time solvable if $\mathcal{G}$ is the class of complete bipartite graphs [23] (see [13] for a faster algorithm).

The STABLE($\mathcal{G}$) problem has also been studied for hereditary graph classes $\mathcal{G}$ with either the property that for all constants $c > 0$, for n sufficiently large, the number of labelled graphs in the class with n vertices is at most n^{cn}, or with the property that there exist constants $c_1, c_2 > 0$ such that, for n sufficiently large, the number of labelled graphs in the class with n vertices is at least $n^{c_1 n}$ and at most $n^{c_2 n}$. (Such classes are said to have, respectively, subfactorial or factorial speed [21,37].)

By relaxing the condition on the set A being independent we obtain the more general problem of $(\mathcal{G}_1, \mathcal{G}_2)$-RECOGNITION, which asks whether the vertex set of a graph can be decomposed into disjoint sets A and B, such that A induces a graph in $\mathcal{G}_1$ and B induces a graph in $\mathcal{G}_2$. For instance, if $\mathcal{G}_1$ is the class of cliques and $\mathcal{G}_2$ is the class of disjoint unions of cliques, then the $(\mathcal{G}_1, \mathcal{G}_2)$-RECOGNITION problem is equivalent to recognizing unipolar graphs (see [40] for a quadratic algorithm). Generalizing STABLE($\mathcal{G}$) can also lead to a family of transversal problems, such as FEEDBACK VERTEX SET. However, such generalizations are beyond the scope of our article.

1.3 | Our results

In Section 2, we consider near-bipartite decompositions of *subcubic* graphs (i.e. graphs of maximum degree at most 3). Recall that by Theorem 1 the only connected subcubic graph that is not near-bipartite is K_4 (see also [49]) and so near-bipartite subcubic graphs can be recognized in polynomial time. As Theorem 1 is concerned with partitions where, additionally, the independent set of the partition is of maximum size its proof is not algorithmic since INDEPENDENT SET for cubic graphs is NP-complete [26]. Neither does the proof of [49] yield a

linear-time algorithm. We will give an $O(n)$-time algorithm that finds a near-bipartite decomposition of any subcubic graph with no connected component isomorphic to K_4.

We say a partition (A, B) of the vertex set of a graph is a *k-degenerate decomposition* if A is independent and B induces a $(k-2)$-degenerate graph, so Section 2 is concerned with 3-degenerate decompositions of graphs of maximum degree 3. In Section 3, we consider, more generally, k-degenerate decompositions of graphs of maximum degree at most k for any $k \geq 3$. By Theorem 5, the only connected graph with maximum degree k that does not have a k-degenerate decomposition is K_{k+1}, but Theorem 5 does not imply a polynomial-time algorithm for finding such a decomposition. We give an $O(n^2)$-time algorithm to find a decomposition for any $k \geq 3$ (in contrast with the $O(n)$-time algorithm in Section 2 for the special case when $k = 3$).

Our results in Sections 2 and 3 provide an algorithmic version of Theorem 5 and, as Theorem 5 generalizes Theorem 2, they also imply an algorithmic version of Theorem 2.

In Section 4 we prove that the problem of deciding whether a graph of maximum degree $2k-2$ has a k-degenerate decomposition is NP-complete. We do this by adapting the proof of the aforementioned result of Yang and Yuan [49], which states that recognizing near-bipartite graphs of maximum degree 4 is NP-complete (the $k = 3$ case). In Section 5 we apply our algorithms from Sections 2 and 3 to completely settle the complexity classification of the graph colouring reconfiguration problem considered in [24]. Finally, in Section 6, we give directions for future work.

2 | A LINEAR TIME ALGORITHM FOR GRAPHS OF MAXIMUM DEGREE AT MOST 3

To prove the result of this section, we need the following terminology. The *claw* is the graph with vertices c, v_1, v_2, v_3 and edges cv_1, cv_2, cv_3; the vertex c is the *centre* of the claw (see Figure 1). The *triangular prism* is the graph obtained from two triangles on vertices u_1, u_2, u_3 and v_1, v_2, v_3, respectively, by adding the edges u_iv_i for $i \in \{1, 2, 3\}$ (see Figure 1). The *diamond* is the graph with vertices v, w, x, y and edges vw, vx, vy, wx, wy (see Figure 3). Two distinct vertices are *false twins* if they have the same neighbourhood (note that such vertices must be nonadjacent).

Theorem 6. *Let G be a subcubic graph on n vertices with no connected component isomorphic to K_4. Then a near-bipartite decomposition of G can be found in $O(n)$ time.*

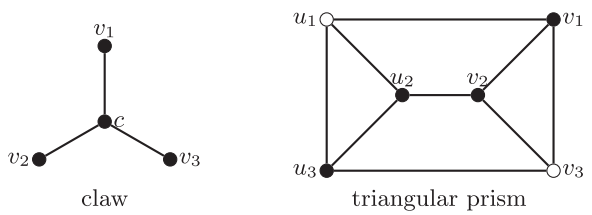


FIGURE 1 The claw and the triangular prism. A near-bipartite decomposition of the triangular prism is indicated: the white vertices form an independent set and the black vertices induce a forest

Proof. We will repeatedly apply a set of *rules* to G. Each rule takes constant time to apply and after each application of a rule, the resulting graph contains fewer vertices. The rules are applied until the empty graph is obtained. We then reconstruct G from the empty graph by working through the rules applied in reverse order. As we rebuild G in this way, we find a near-bipartite decomposition of each obtained graph. We do this by describing how to extend, in constant time, a near-bipartite decomposition of a graph before some rule is undone to a near-bipartite decomposition of the resulting graph after that rule is undone. If we can do this then we say that the rule is *safe*. We conclude that the total running time of the algorithm is $O(n)$. It only remains to describe the rules, show that it takes constant time to do and undo each of them and prove that they are safe.

Let u be an arbitrary vertex of G. Our choice of u as an arbitrary vertex implies that u can be found in constant time. We then use the first of the following rules that is applicable.

Rule 1. If there is a vertex v of degree at most 2 that is at distance at most 3 from u, then remove v. (Note that $v = u$ is possible; a similar comment can be made for some of the other rules as well.)

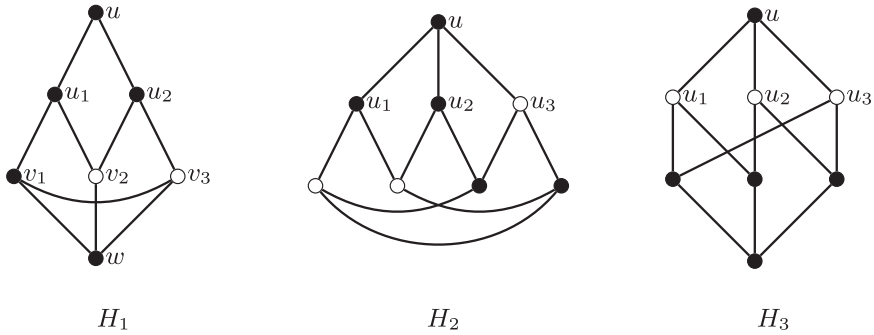
Rule 2. If G contains an induced diamond D whose vertices are at distance at most 3 from u, then remove the vertices of D.

Rule 3. If there is a pair of false twins u_1, u_2 each at distance at most 2 from u, then remove u_1, u_2 and their neighbours.

Rule 4. If u is in a connected component that is a triangular prism P, then remove the vertices of P.

Rule 5. If Rules 1–4 do not apply but u is in a triangle T, then the neighbours of the vertices in T that are outside T are pairwise distinct (since there is no induced diamond or K_4) and at least two of them, which we denote by x', y', are non-adjacent (otherwise u belongs to a triangular prism). Remove the vertices of T and add an edge between x' and y'.

Rule 6. If u is the centre of an induced claw but has a neighbour v that belongs to a triangle, then apply one of the Rules 1–5 on v.



H_1 H_2 H_3

FIGURE 2 The graphs used in Rule 7. A near-bipartite decomposition of each is indicated: the white vertices form an independent set and the black vertices induce a forest

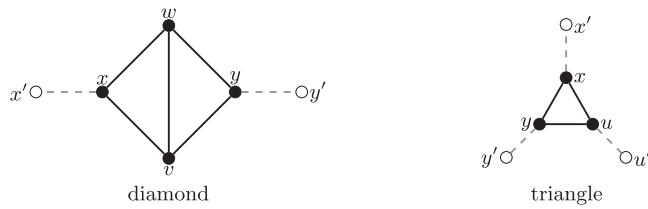


FIGURE 3 The diamond and triangle (solid edges and vertices) together with their neighbourhoods in a cubic graph

Rule 7. If the graph induced by the vertices at distance at most 3 from u contains one of the graphs H_1, H_2 or H_3, depicted in Figure 2, with the vertex u in the position shown in the figure, then remove the vertices of this graph H_i.

Rule 8. If Rules 1–7 do not apply but u is the centre of an induced claw and its three neighbours u_1, u_2, u_3 are also centres of induced claws, then remove u, u_1, u_2, u_3 and for $i \in \{1, 2, 3\}$ add an edge joining the two neighbours of u_i distinct from u and denote it by e_i; we say that such an edge is *new* (note that such neighbours of two distinct u_i and u_j may overlap).

Let us show that at least one of the rules is always applicable. Suppose that, on the contrary, there is a vertex u of a subcubic graph for which no rule applies. Then u and its neighbours each have degree 3 (Rule 1) and so each either belongs to a triangle or is the centre of an induced claw. By Rule 5, u must be the centre of an induced claw and therefore, by Rule 6, the same is also true for each neighbour of u. This implies that Rule 8 applies, a contradiction.

Because G is subcubic, each of these rules takes constant time to verify and process. In particular, in some rules we need to detect some induced subgraph of constant size that contains u or replace u by some other vertex v. In all such cases we need to explore a set of vertices of distance at most 4 from u. As G is subcubic, this set has size at most $1 + 3 + 3^2 + 3^3 + 3^4 = 121$, so we can indeed do this in constant time.

It is clear that, as claimed, the application of a rule reduces the number of vertices and that if we repeatedly choose an arbitrary vertex u and apply a rule, we eventually obtain the empty graph. We now consider undoing the applied rules in reverse order to rebuild G. As this is done, we will irrevocably *colour* vertices with colour 1 or 2 in such a way that the vertices coloured 1 will form an independent set and the vertices coloured 2 will induce a forest. Thus a rule is safe if this colouring can be extended whenever that rule is undone. When we reach G, the final colouring will correspond to the required near-bipartite decomposition.

We must prove each rule is safe. At each step of reconstructing G, we refer to the graph before a rule is undone as the *subsequent graph* and to the graph after that rule is undone as the *prior graph*. Note that the application of any of the Rules 1–8 again yields a subcubic graph. By the result of Yang and Yuan [49], every connected subcubic graph is near-bipartite, apart from K_4. So we need to ensure that an application of a rule does not create a K_4. This cannot happen when we remove vertices, but we will need to consider it for Rules 5 and 8.

Claim 1. Rules 1–4 are safe.

In Rules 1–4 we only delete vertices. Rule 1 is safe since if both neighbours of v are coloured 2, then v can be coloured 1; otherwise v can be coloured 2. We note that from hereon we can assume that all vertices at distance at most 3 from u have degree 3.

To see that Rule 2 is safe, let D be the diamond with vertex labels as illustrated in Figure 3. If x' and y' are coloured 2, we colour x and y with 1 and v and w with 2. Otherwise we colour v with 1 and x, y and w with 2.

We now show that Rule 3 is safe. As G is subcubic, every vertex in $N_G(u_1)$ with a neighbour in $N_G(u_1)$ has no neighbours outside $N_G(u_1) \cup \{u_1, u_2\}$ and every vertex in $N_G(u_1)$ with no neighbour in $N_G(u_1)$ has at most one neighbour not equal to u_1 or u_2. Moreover, as G is subcubic, $N_G(u_1)$ contains no cycle. Hence we can always colour u_1, u_2 with 1 and the vertices of $N_G(u_1)$ with 2 regardless of the colours of vertices outside $N_G(u_1) \cup \{u_1, u_2\}$. Indeed, every vertex of $N_G(u_1)$ will have at most one neighbour that is not coloured 1, so cannot be in a cycle of vertices coloured 2 in the prior graph.

Rule 4 is also safe since P is 3-regular and hence would be a connected component of the prior graph, so we can colour its vertices by assigning colour 1 to exactly one vertex from each of the two triangles (which are nonadjacent) and colour 2 to its other vertices (see Figure 1). This completes the proof of Claim 1.

Claim 2. Rules 5 and 6 are safe.

First, let us demonstrate that Rule 5 is safe. If x' and y' are contained in a K_4 of the subsequent graph, then the prior graph contains a diamond whose vertices are at distance at most 3 from u. This contradicts Rule 2. Let T be the triangle with vertex labels as illustrated in Figure 3. Suppose x', y' and u' are coloured 2. Then we colour u with 1 and x and y with 2. The vertices in the prior graph with colour 2 still induce a forest, as we have replaced an edge in the forest by a path on four vertices. Suppose x' and y' are coloured 2 and u' is coloured 1. Then we colour x with 1, and y and u with 2. Otherwise, since x' and y' are joined by an edge in the subsequent graph, we may assume that x' has colour 1 and y' has colour 2. In this case we can colour y with 1, and x and u with 2. This completes the proof that Rule 5 is safe. Since Rules 1–5 are safe, it follows that Rule 6 is also safe. This completes the proof of Claim 2.

Claim 3. Rule 7 is safe.

We now show that Rule 7 is safe. Suppose u is contained in H_1. We use the vertex labels from Figure 2. As Rule 1 could not be applied, we find that u has a third neighbour u_3 distinct from u_1 and u_2. Regardless of whether u_3 is coloured 1 or 2, we colour u, u_1, u_2, v_1, w with 2 and v_2, v_3 with 1 to obtain a near-bipartite decomposition of G. We can also readily colour the vertices of H_2 or H_3 should u be contained in one of them (note that since H_2 and H_3 are 3-regular, these graphs can only appear as connected components in our prior graph). This completes the proof of Claim 3.

Claim 4. Rule 8 is safe.

Suppose that the subsequent graph contains fewer than three new edges. Then we may assume without loss of generality that $e_1 = e_2$. Then u_1 and u_2 are false twins at

distance 1 from u and we can apply Rule 3, a contradiction. So we may assume that the subsequent graph contains exactly three new edges.

We claim that the application of Rule 8 does not yield a K_4. For contradiction, suppose it does. Let K be the created K_4. Then at least one new edge is contained in K. If exactly one new edge e is contained in K, then $K - e$ is a diamond in the prior graph. Then we could have applied Rule 2, a contradiction. If all three new edges are in K, then they must induce either a path on four vertices, a triangle or a claw in the prior graph. In the first case the prior graph is H_2, in the second case the prior graph is H_3, and in the third case u and the centre of the claw are false twins. Thus we would have applied Rule 7 or Rule 3, a contradiction. Finally, suppose that K contains exactly two new edges, say e_1 and e_2. If e_1 and e_2 do not share a vertex, then they cover the vertices of K. Hence the end-vertices of e_1 are false twins (at distance 2 from u) in the prior graph, since they are both adjacent to u_1 and to each end-vertex of e_2. Then we could have applied Rule 3, a contradiction. If $e_1 = v_1v_2$ and $e_2 = v_3v_4$ share a vertex, say $v_2 = v_4$, then v_1 and v_3 are adjacent in the prior graph and the vertex $w \in K\backslash\{v_1, v_2, v_3\}$ is adjacent only to v_1, v_2 and v_3. Therefore Rule 7 could have been applied, a contradiction.

Thus an application of Rule 8 does not yield a K_4. We note that the three new edges are distinct else two of u_1, u_2, u_3 would be false twins and Rule 3 would apply. We may colour u_1, u_2, u_3 with 2 and u with 1. This will yield a near-bipartite decomposition since even if the two end-vertices of a new edge are coloured 2, in the prior graph the vertices coloured 2 will still induce a forest, in which such a new edge is replaced by a path of length 2 (and as the new edges are distinct each one will be replaced by exactly one path of length 2). This completes the proof of Claim 4 and therefore completes the proof of Theorem 6. □

3 | A QUADRATIC TIME ALGORITHM FOR GRAPHS OF BOUNDED MAXIMUM DEGREE

Let $k \geq 3$ be an integer. Recall that a graph G has a k-degenerate decomposition if its vertex set can be decomposed into sets A and B where A is an independent set and B induces a $(k-2)$-degenerate graph. Note that 3-degenerate decompositions are near-bipartite decompositions. In this section we give an $O(n^2)$ algorithm (Algorithm 1) for finding a k-degenerate decomposition of a graph on n vertices of maximum degree at most k for every $k \geq 3$. (Note that for $k = 3$ we can also use Theorem 6.)

Algorithm 1 is stated below. In outline it is very simple: one picks a pair of nonadjacent vertices with a common neighbour and considers the input graph with this pair removed. If this altered graph is connected, then it is straightforward to find a k-degenerate decomposition (Lemma 2). Otherwise one considers one arbitrarily chosen connected component C of the altered graph: if C has a simple structure, then again the decomposition can be found (Lemmas 3–5). Otherwise a further pair of nonadjacent vertices with a common neighbour is found within C and the algorithm recurses. Before we consider these cases, we need some further definitions and a short technical lemma (Lemma 1).

For $k \geq 1$, we say that an order $v_1, v_2, ..., v_n$ of the vertices of a graph G is *k-degenerate* if for all $i \geq 2$, the vertex v_i has at most k neighbours in $\{v_1, ..., v_{i-1}\}$. It is clear that a graph is k-degenerate if and only if it has a k-degenerate order. If $\mathcal{O}$ is a k-degenerate order for G, and W is a subset of the vertex set of G, then we let $\mathcal{O}|_W$ be the restriction of $\mathcal{O}$ to W, and let $G[W]$ be

the subgraph of G induced by W. For a set of vertices $S \subseteq V$, we denote the *neighbourhood* of S by $N_G(S) = \bigcup\{N(u) | u \in S\} \backslash S$.

We need the following lemma, which is a refinement of Lemma 8 in [24] with the same proof. We include a sketch of the argument for completeness.

Lemma 1. *Let $k \geq 2$. Let G be a $(k-1)$-degenerate graph on n vertices. If a $(k-1)$-degenerate order $\mathcal{O}$ of G is given as input, a k-degenerate decomposition (A, B) of G can be found in $O(kn)$ time. In addition, we can ensure that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order of $G[B]$, the set A is a maximal independent set and the first vertex in $\mathcal{O}$ belongs to A.*

Proof. Let $\mathcal{O}$ be $v_1, v_2, ..., v_n$. Consider the greedy algorithm that starts with two empty sets A and B, and, at step i, assigns v_i to A unless v_i has a neighbour of smaller index already in A, in which case it assigns v_i to B. Clearly the set A is an independent set and every vertex of B has at least one neighbour in A, so A is a maximal independent set. At any step i, if the vertex v_i is assigned to B, then it has a neighbour of smaller index that belongs to A. This implies that v_i has at most $k-2$ neighbours of smaller index that belong to B. □

A pair of distinct nonadjacent vertices $\{u, v\}$ in a graph G is *strong* if u and v have a common neighbour in each connected component of the graph $G \backslash \{u, v\}$. In particular, note that if u and v have a common neighbour and $G \backslash \{u, v\}$ is connected, then $\{u, v\}$ is a strong pair.

Lemma 2. *Let $k \geq 3$. Let G be a connected k-regular graph on n vertices that contains a strong pair $\{u, v\}$. If $\{u, v\}$ is given as input, a k-degenerate decomposition of G can be found in $O(kn)$ time.*

Proof. Let G' be the graph obtained by identifying u and v into a new vertex z with $N_G(z) = N_G(u) \cup N_G(v)$. For each connected component C of $G' \backslash \{z\}$, let z_C be a vertex of C that is adjacent to both u and v in G. We find a $(k-1)$-degenerate order $\mathcal{O}$ of the vertices of G' as follows. Let z be the first vertex in $\mathcal{O}$. Consider each connected component C of $G' \backslash \{z\}$ in turn, and append to $\mathcal{O}$ the vertices of C in the reverse of the order they are found in a breadth-first search from z_C (note that this can be done in $O(kn)$ time). Then $\mathcal{O}$ is a $(k-1)$-degenerate order as every vertex has a neighbour later in the order except for each z_C vertex, which has degree $k-1$. It follows from Lemma 1 that we can find a k-degenerate decomposition (A, B) of G' with $z \in A$. Then $(A \backslash \{z\} \cup \{u, v\}, B)$ is a k-degenerate decomposition of G as $G[B] = G'[B]$, and u and v are nonadjacent so $A \backslash \{z\} \cup \{u, v\}$ is an independent set. □

Lemma 3. *Let $k \geq 3$. Let G be a k-regular connected graph on n vertices, which contains a set S of $k+1$ vertices that induces a clique minus an edge uv. If S, u and v are given as input, then a k-degenerate decomposition of G can be found in $O(kn)$ time.*

Proof. Let x be a vertex in S distinct from u and v. Let G' be the graph obtained from G by deleting S. Each of u and v has exactly one neighbour that does not belong to S and all other vertices of S have no neighbours outside S. Let t be the neighbour of u not in S, and let w be the neighbour of v not in S. We may assume that t is distinct from w, otherwise we are done by Lemma 2. We can find a $(k-1)$-degenerate order $\mathcal{O}$ of G' in $O(kn)$ time

by taking the vertices in the reverse of the order they are found in a breadth-first search from t, and then, if t and w do not belong to the same connected component of G', appending the vertices in the reverse of the order they are found in a breadth-first search from w. By Lemma 1, we can compute a k-degenerate decomposition (A, B) of G' in $O(kn)$ time such that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order of B. If both t and w belong to B, let $A' = A \cup \{u, v\}$ and $B' = B \cup (S \backslash \{u, v\})$ and, since $(S \backslash \{u, v\})$ is a clique on $k-1$ vertices with no edge joining it to B, if follows that (A', B') is a k-degenerate decomposition of G. Assume now without loss of generality that $t \in A$ (we make no assumption about whether w is also in A). Then let $A' = A \cup \{x\}$ and $B' = B \cup (S \backslash \{x\})$. Then A' is an independent set. Recall that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order of B. We show that we can append the vertices of $S \backslash \{x\}$ to obtain a $(k-2)$-degenerate order of B'. First add v, then the vertices of $S \backslash \{u, v, x\}$ and finally u. It is clear that no vertex has more than $k-2$ neighbours earlier in the order. □

Lemma 4. *Let $k \geq 3$. Let G be a k-regular connected graph on n vertices containing a clique K on k vertices whose neighbourhood is of size 2. If K is given as input, a k-degenerate decomposition of G can be found in $O(kn)$ time.*

Proof. Let u and v be vertices not in K such that for each vertex in K, its unique neighbour not in K is either u or v. Neither u nor v can be adjacent to every vertex in K (as then the other would be adjacent to none, contradicting the premise that the neighbourhood has size 2). Since $k \geq 3$, one of u and v has at least two neighbours in K. Consider the graph G' obtained from G by removing K and adding the edge uv (if it does not already exist). Note that u and v each have degree at most k in G' and at least one of them, say u, has degree less than k. Therefore, we can find a $(k-1)$-degenerate order $\mathcal{O}$ of G' in $O(kn)$ time by taking the vertices in the reverse of the order they are found in a breadth-first search from u. Thus we can obtain a k-degenerate decomposition (A, B) of G' by Lemma 1, such that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order of $G[B]$ and A is a maximal independent set. At least one of u and v must belong to B. Assume without loss of generality that either $v \in A, u \in B$ or both u and v belong to B, and, in the latter case, assume that u has at least two neighbours in K. Consider a neighbour t of u in K. We set $A' = A \cup \{t\}$ and $B' = B \cup (K \backslash \{t\})$, and claim that (A', B') is a k-degenerate decomposition of G. It is clear that A' is an independent set since no vertex in K, including t, is adjacent to both u and v. Recall that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order for $G[B]$. We must amend it to find a $(k-2)$-degenerate order for $G[B']$ that also includes the vertices of $K \backslash \{t\}$. We consider two cases.

First suppose $v \in A$. Then append to $\mathcal{O}|_B$ first the neighbours of u in $K \backslash \{t\}$ and then the neighbours of v. As the vertices of $K \backslash \{t\}$ are adjacent to t the only one that could have more than $k-2$ vertices before it in the order is the one that appears last, but this is also adjacent to v so we do indeed have a $(k-2)$-degenerate order.

Now suppose $v \in B$. Then u has a neighbour in G' that belongs to A (as A is a maximal independent set). Hence u has at most $k-2$ neighbours in B'. Append to $\mathcal{O}|_B$ the vertices of $K \backslash \{t\}$, ending with a neighbour of u (we know there is at least one), then move u to be the last vertex in the order. Again the only vertex of $K \backslash \{t\}$ that could have more than $k-2$ neighbours before it in the order is the one that appears latest in the order, and by choosing it to be a neighbour of u and putting u last in the order we ensure that a $(k-2)$-degenerate order is obtained. □

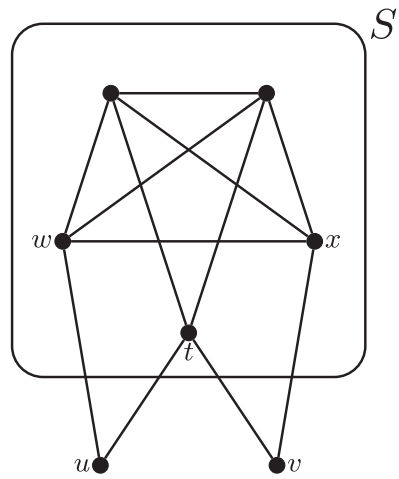


FIGURE 4 An example of a (u, v)-lock S with special vertices $(t, \{w, x\})$, where $|S| = 5$. Note that w and x each have exactly one neighbour in $\{u, v\}$

Given a graph G, five of its vertices t, u, v, w, x and a set of vertices S, we say that S induces a (u, v)-*lock with special vertices*$(t, \{w, x\})$ if $t, w, x \in S$ and $N_G(S) = \{u, v\}$, and both u and v are adjacent to t, each vertex in $\{w, x\}$ is adjacent to precisely one vertex in $\{u, v\}$, and $G[S]$ contains all possible edges except for wt and xt (see Figure 4). We say that S induces a *lock* if it induces a (u, v)-*lock with special vertices*$(t, \{w, x\})$ for some choice of t, u, v, w, x.

Lemma 5. *Let $k \geq 3$. Let G be a k-regular connected graph on n vertices containing a (u, v)-lock S with special vertices $(t, \{w, x\})$. If S and u, v, t, w, x are given as input, then a k-degenerate decomposition of G can be found in $O(kn)$ time.*

Proof. Since t has two neighbours outside S, it has $k - 2$ neighbours in S. As S also contains w and x (which are not neighbours of t), it follows that $S\backslash\{t\}$ is a clique on k vertices. If w and x have the same neighbour in $\{u, v\}$, say u, then $N_G(S\backslash\{t\}) = \{t, u\}$ and so we are done by Lemma 4. We may therefore assume that w and x have distinct neighbours in $\{u, v\}$. Let G' be the graph obtained from G by deleting $S\backslash\{t\}$ and note that G' is connected since t is adjacent to both u and v. Note that both u and v have degree $k - 1$ in G'. We can therefore find a $(k - 1)$-degenerate order $\mathcal{O}$ of G' in $O(kn)$ time by taking the vertices in the reverse of the order they are found in a breadth-first search from u. Furthermore, since the only neighbours of t in G' are u and v, both of which have degree $k - 1$, by moving t to the start of the order $\mathcal{O}$, we obtain a another $(k - 1)$-degenerate order $\mathcal{O}'$. By Lemma 1, we can therefore find a k-degenerate decomposition (A, B) of G' such that $\mathcal{O}' \mid_B$ is a $(k - 2)$-degenerate order on B and $t \in A$. Thus both u and v belong to B. We let $A' = A \cup \{w\}$ and $B' = B \cup (S\backslash\{w, t\})$, and claim that (A', B') is a k-degenerate decomposition of G. It is clear that A' is an independent set. We have the $(k - 2)$-degenerate order $\mathcal{O}' \mid_B$ on B. We obtain a $(k - 2)$-degenerate order on B' by appending to $\mathcal{O}' \mid_B$ the vertices of $S\backslash\{w, t\}$ beginning with x. Indeed, the only neighbour of x that is earlier in the order is its single neighbour in $\{u, v\}$ (and note that $1 \leq k - 2$ since $k \geq 3$). Furthermore, since $w, t \in A'$, every vertex in $S\backslash\{w, t, x\}$ has only $k - 2$ neighbours in B'. □

A pair of nonadjacent vertices u, v in a graph is a *good* pair if u and v have a common neighbour. Note that if a good pair u, v is not strong, then $G\backslash\{u, v\}$ must be disconnected. We are now ready to state and prove the following result.

Theorem 7. *Let $k \geq 3$ and G be a graph on n vertices with maximum degree at most k. If no connected component of G is isomorphic to K_{k+1}, then a k-degenerate decomposition of G can be found in $O(k^2n^2)$ time.*

Proof. We may assume that G is connected, otherwise it can be considered componentwise. If G is not k-regular, then it has a vertex u of degree at most $k-1$, so we can find a $(k-1)$-degenerate order $\mathcal{O}$ of G in $O(kn)$ time by taking the vertices in the reverse of the order they are found in a breadth-first search from u. In this case, we are done by Lemma 1. For k-regular graphs, we use the procedure shown in Algorithm 1 below. (We note that if G is biconnected then, by [2, Lemma 3], there is always a good pair u, v such that $G\backslash\{u, v\}$ is connected, but this does not aid us in finding an algorithm for general graphs.)

Algorithm 1: Finding a k-degenerate decomposition for connected k-regular graphs.

Input : A connected k-regular graph G
Output : A k-degenerate decomposition of G

1 *find a good pair u, v and let C be a connected component of $G \setminus \{u, v\}$*;
2 **if** *u, v is a strong pair* **then** apply Lemma 2;
3 **else if** *the union of C and one or both of u and v is a clique on $k+1$ vertices minus an edge* **then** apply Lemma 3;
4 **else if** *C is a clique on k vertices whose neighbourhood is $\{u, v\}$* **then** apply Lemma 4;
5 **else if** *C is a (u, v)-lock* **then** apply Lemma 5;
6 **else**
7 | *find a good pair $u', v' \in C$ such that either $C' = G \setminus \{u', v'\}$ is connected or $G \setminus \{u', v'\}$ has a connected component C' that is strictly contained in C*;
8 | set $u \leftarrow u'$, $v \leftarrow v'$, $C \leftarrow C'$;
9 | **go to** *Line 2*
10 **end**

Let us make a few comments on Algorithm 1. As G is regular, connected and not complete, we can initially choose any vertex as u and find another vertex v to form a good pair in $O(k^2) = O(kn)$ time. If we perform a breadth-first search (which takes $O(n+m) = O(kn)$ time) from a neighbour of u that retreats from u or v whenever they are encountered, we discover a connected component of $G\backslash\{u, v\}$. If the connected component contains a common neighbour of u and v but is not equal to $G\backslash\{u, v\}$, we repeat starting from a neighbour of u or v that was not discovered. Thus we discover in $O(k^2n)$ time that either u, v is a strong pair (if we find all connected components of $G\backslash\{u, v\}$ and they each contain a common neighbour of u and v), or that it is not. We set C to be one of the connected components of $G\backslash\{u, v\}$ arbitrarily. By Lemma 2, we therefore conclude that Lines 1 and 2 take $O(k^2n)$ time. It is easy to check in $O(kn)$ time whether we apply Lemmas 3–5 on Lines 3–5 and applying these lemmas takes $O(kn)$ in each case. Now suppose that we do not apply any of these lemmas, in which case we reach Line 7. We will show that we can find u', v' and, if necessary, C' in $O(kn)$ time. If we find u', v' such that $C' = G\backslash\{u', v'\}$ is connected, then u', v' is a strong pair, so after executing Line 9, the algorithm will stop on Line 2. In all other cases C' will be strictly smaller than C. This means that we apply Line 9 at most $O(n)$ times, implying that we

execute Lines 2–9 at most $O(n)$ times. This will give an overall running time of $O(k^2n^2)$. It remains to show that if execution reaches Line 7 then we can find the required good pair u', v' and the connected component C' in $O(kn)$ time.

Let us first show that C contains good pairs—that is, that it is not a clique. If C is a clique, then it contains either $k-1$ or k vertices (as each vertex has degree k in G and the only other possible neighbours of vertices in C are u and v). If C has $k-1$ vertices, then each vertex of C must be adjacent to both u and v and we would have applied Lemma 3 on Line 3, a contradiction. If C has k vertices, then each vertex of C is adjacent to exactly one of u and v (and neither u nor v can be adjacent to every vertex in C, as this would form a K_{k+1}, contradicting the fact that G is connected), in which case we would have applied Lemma 4 on Line 4, a contradiction. Therefore we may assume that C is not a clique.

We need to describe how to choose a good pair u', v' in C. If we can show that u and v are in the same connected component of $G\backslash\{u', v'\}$ (which must necessarily contain all of $G\backslash C$), then we are done as either $G\backslash\{u', v'\}$ is connected or there is another connected component C' of $G\backslash\{u', v'\}$ which must be contained in C (and note that in this case C' can be found in $O(kn)$ time using breadth-first search).

If u and v have a common neighbour outside C or at least three common neighbours in C, then any good pair in C can be chosen as u', v' (as u and v will then be in the same connected component of $G\backslash\{u', v'\}$ as they have at least one common neighbour i.e. not one of u', v'). If u and v have exactly two common neighbours t_1, t_2 that both belong to C, then any good pair other than t_1, t_2 can be chosen as u', v'. If t_1, t_2 is the only good pair in C (so all other vertices in C are adjacent), then C is a clique minus an edge and must contain k vertices (t_1, t_2 and the $k-2$ neighbours of t_1 that are not in $\{u, v\}$). Considering degree, any vertex in C other than t_1 or t_2 must be adjacent to exactly one of u and v. If every vertex in $C\backslash\{t_1, t_2\}$ is adjacent to, say u, then $C \cup \{u\}$ is a clique on $k+1$ vertices minus an edge and we would have applied Lemma 3 on Line 3, a contradiction. We may therefore assume that at least one vertex in $C\backslash\{t_1, t_2\}$ is adjacent to u and at least one is adjacent to v, so there is a path from u to v avoiding t_1 and t_2, and $G\backslash\{t_1, t_2\}$ is connected, so we are done.

Finally, suppose that u and v have exactly one common neighbour t that belongs to C. Then any good pair not including t can be chosen as u', v', as then u and v will be in the same connected component of $G\backslash\{u', v'\}$. Suppose, for contradiction, that no such pair exists. Then $C\backslash\{t\}$ is a clique. The vertex t has $k-2$ neighbours in $C\backslash\{t\}$. Since $k \geq 3$, let z be one of those neighbours. Since t is the only common neighbour of u and v, we have that z can only be adjacent to at most one of u and v. Therefore, t has a neighbour nonadjacent to z, so z must have a neighbour nonadjacent to t, which we denote w. As w is also adjacent to at most one of u and v, it also has a neighbour x that is a non-neighbour of t (and cannot be t itself). So $C\backslash\{t\}$ contains at least k vertices: the $k-2$ neighbours of t plus w and x. As $C\backslash\{t\}$ induces a clique, it must have exactly k vertices, since G cannot contain a K_{k+1}. Thus the set C forms a lock, and so we would have applied Lemma 5 on Line 5. This contradiction completes the proof. □

Theorems 6 and 7 provide an algorithmic version of Theorem 5. Moreover, Theorems 6 and 7 concern decompositions (A, B) of the vertex set of a graph where A is independent and B induces a $(k-2)$-degenerate graph. As B therefore cannot be a clique on k vertices, both theorems also provide an algorithmic version of Theorem 2. As previously discussed, in these algorithmic theorems we have necessarily removed the requirement that the independent set A have maximum size.

4 | HARDNESS FOR GRAPHS OF LARGER MAXIMUM DEGREE

In this section, we prove that for every $k \geq 3$, the problem of deciding whether a graph of maximum degree $2k-2$ has a k-degenerate decomposition is NP-complete.

The $k = 3$ case is the problem of deciding whether a graph of maximum degree 4 is near-bipartite and was proven by Yang and Yuan [49, Theorem 3.5]. Our generalization adapts their proof in a straightforward way. We first need some definitions.

Let $k \geq 3$ and $p \geq 1$ be integers. We will construct a graph H_k^p to be used in the NP-completeness reduction. The vertices of H_k^p are partitioned into $2p+1$ *levels*. Level 0 contains a single vertex. For $1 \leq i \leq 2p-1$, if i is odd, then, for each vertex u in level $i-1$, level i contains a distinct set of $k-1$ vertices which induce a clique, which we call an *odd-level clique*, and are each joined by an edge to u, which is said to be their *parent*. For $2 \leq i \leq 2p$, if i is even, then for each odd-level clique K in level $i-1$, level i contains a distinct set of $k-1$ pairwise non-adjacent vertices which are each adjacent to every vertex of K. We refer to level i as an *odd* or *even* level according to the value of i. We refer to the single vertex in level 0 as the *foot* of the graph. We refer to level $2p$ as the *top* level. We note that the foot and each vertex in the top level has degree $k-1$, and that every other vertex in H_k^p has degree $2k-2$. We call each collection of $k-1$ vertices in the top level that have the same neighbourhood a *top set*, and note that there are $(k-1)^{p-1}$ top sets. See Figure 5 for an illustration of H_3^3.

Given such a graph H_k^p, let A_t be a set of vertices that contains one vertex from each odd-level clique. Let $B_t = V(H_k^p) \backslash A_t$. Let A_f be the set of all vertices in even levels, and let B_f be the set of all vertices in odd levels. We say that (A_t, B_t) is a *true* decomposition of H_k^p, and that (A_f, B_f) is a *false* decomposition of H_k^p. It is easy to check that A_t and A_f are independent sets. We can see that B_t and B_f induce $(k-2)$-degenerate graphs by considering the vertices ordered by level, starting from 0. These are $(k-2)$-degenerate orders since, in each of B_t and B_f, each vertex in the set from level i has at most $k-2$ neighbours in the set that belong to level i or level $i-1$. Thus true and false decompositions are k-degenerate decompositions. The next lemma shows that every k-degenerate decomposition is one of these two types.

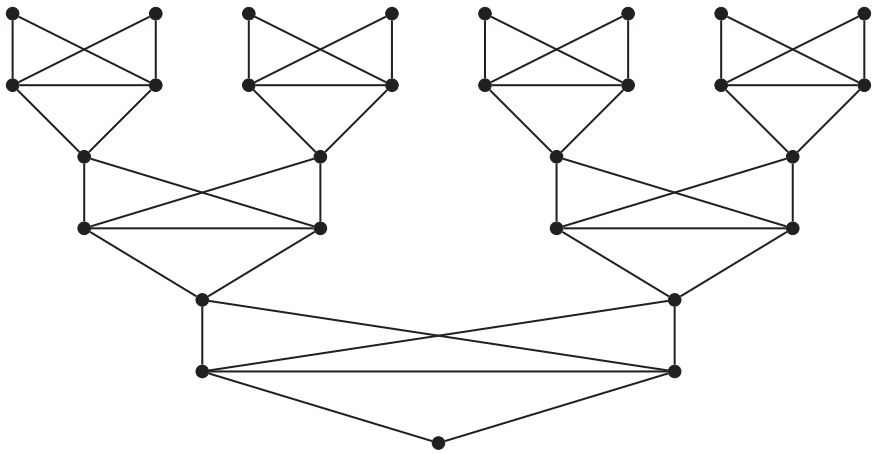


FIGURE 5 The graph H_3^3

Lemma 6. *Let $k \geq 3$ and $p \geq 1$ be integers. Let (A, B) be a k-degenerate decomposition of H_k^p. Then (A, B) is either a true decomposition or a false decomposition.*

Proof. Suppose that the foot of H_k^p is in the $(k-2)$-degenerate graph B. We must show that we have a true decomposition. We will first prove, by induction, that the vertices of level i are in B if i is even, and that if i is odd, then exactly one vertex from each of the odd-level cliques is in B. For $i = 0$, this follows by assumption. Now suppose that $i \geq 1$ and i is odd. Then each odd-level clique K in level i has a parent in level $i-1$ that is in B. Thus at least one vertex in K must be in A, otherwise B would contain a clique on k vertices and would not be $(k-2)$-degenerate (and clearly there cannot be more than one vertex of K in A). Finally suppose that $i \geq 2$ and that i is even. Then each vertex v in level i is adjacent to a vertex in level $i-1$ that is in A. Thus v must be in B, otherwise A is not an independent set.

Suppose that the foot of H_k^p is in A. We must show that we have a false decomposition. We will prove, by induction, that the vertices of level i are in A if i is even and in B if i is odd. For $i = 0$, this follows by assumption. Now suppose that $i \geq 1$ and i is odd. Then each vertex v in level i has a parent in level $i-1$ that is in A. Thus v must be in B. Finally suppose that $i \geq 2$ and i is even. Then each vertex v in level i is adjacent to each vertex in a clique on $k-1$ vertices in level $i-1$ that is in B. Thus v must be in A, otherwise B would contain a clique on k vertices and would not be $(k-2)$-degenerate. □

We are now ready to prove our hardness result.

Theorem 8. *For every $k \geq 3$, the problem of deciding whether a graph of maximum degree $2k-2$ has a k-degenerate decomposition is* NP*-complete.*

Proof. The problem is readily seen to belong to NP. We will use a reduction from 1-IN-k-SAT with positive literals only. An instance of this problem is a set of variables $X = \{x_1, ..., x_n\}$ and a set of clauses $\mathcal{C} = \{C_1, ..., C_m\}$ such that each clause C_i is of the form $(x_{i_1} \vee \cdots \vee x_{i_k})$ with $1 \leq x_{i_1} < \cdots < x_{i_k} \leq n$. The question is whether there exists a truth assignment that makes exactly one variable in each clause true, which we call a *good* truth assignment. The problem 1-IN-3-SAT with positive literals only is well known to be NP-complete (see Garey and Johnson [25]) and it is easy to find a reduction to 1-IN-k-SAT with positive literals only.

Given an instance $(X, \mathcal{C})$ of 1-IN-k-SAT, we construct a graph G as follows. Let $p = 1 + \lceil \log_{k-1} m \rceil$ and note that H_k^p has at least m top sets (and the number of occurrences of each variable in $\mathcal{C}$ is at most m). For each variable x_i, let G_i be a copy of H_k^p; we call these graphs *variable gadgets*. For each clause C_j, let F_j be a clique on k vertices; we call these graphs *clause gadgets*. Let each vertex in F_j represent a distinct variable in C_j: if a vertex in F_j represents the variable x_i, then we label it x_i^j. Let G be the disjoint union of the variable and clause gadgets plus the following set of edges: for each variable x_i in each clause C_j, select a distinct top set in G_i and add an edge from each vertex in the top set to x_i^j.

The degree in G of each vertex in each variable gadget is as in H_k^p except that a vertex in a top set might have an additional neighbour in a clause gadget and so have degree k. The degree of each vertex in each clause gadget is $2k-2$, Hence, the maximum degree of

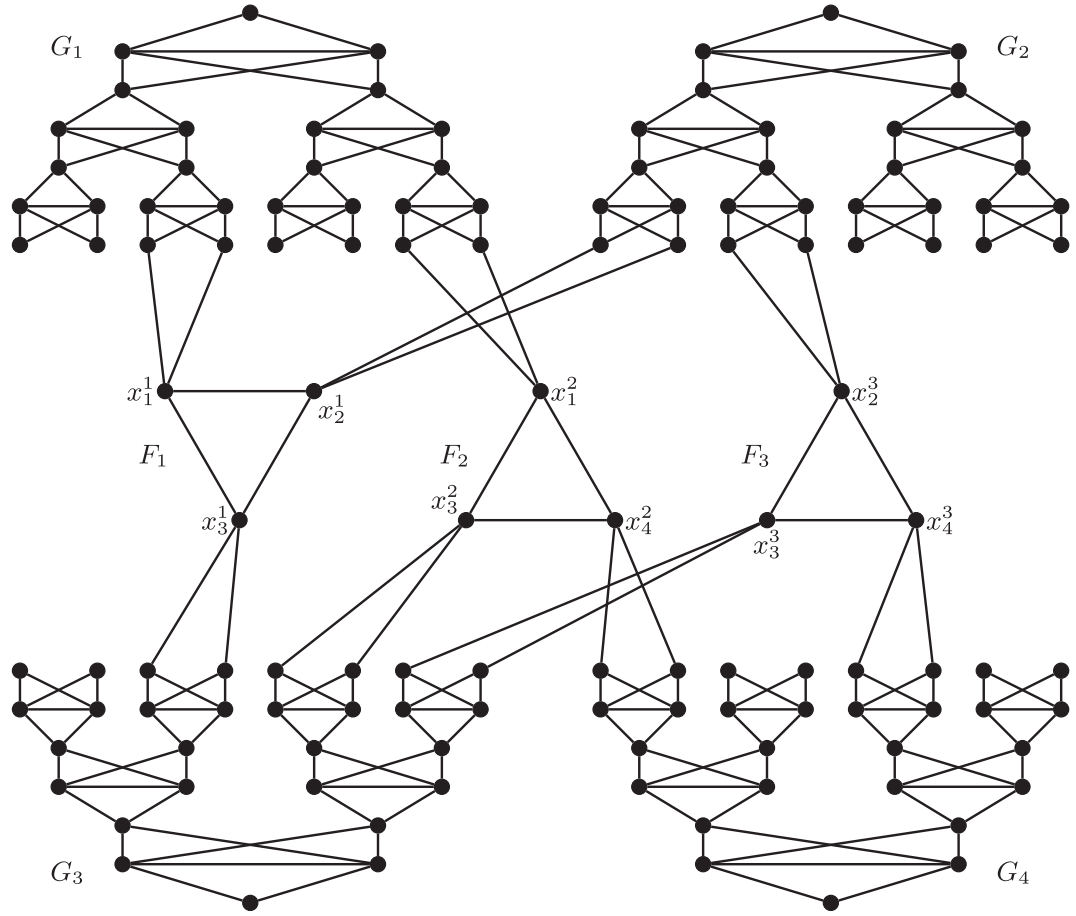


FIGURE 6 The graph G constructed from an instance of 1-IN-3-SAT with $\mathcal{C} = ((x_1 \vee x_2 \vee x_3), (x_1 \vee x_3 \vee x_4), (x_2 \vee x_3 \vee x_4))$

G is $2k - 2$. See Figure 6 for an example. We claim that there is a good truth assignment if and only if G has a k-degenerate decomposition.

First suppose that there is a good truth assignment. We will find a k-degenerate decomposition (A, B) of G. We add the vertices of each variable gadget G_i to A and B in such a way that, on G_i, (A, B) induces a true or false decomposition if x_i is true or false, respectively. A vertex x_i^j in a clause gadget is added to A if x_i is true and is otherwise added to B. We must show that A is an independent set. As the restriction of A to the variable gadgets is an independent set, we only need to show that if a vertex x_i^j in a clause gadget is added to A, then it has no neighbour in A.

As only one variable in C_j is true, x_i^j is the only vertex from F_j in A (and x_i^j has no neighbours in any other clause gadget). The neighbours of x_i^j in G_i are in a top set and so in an even level. By the definition of a true decomposition, this implies that they are all in B. We must show that B is $(k - 2)$-degenerate. By taking orders of the variable gadgets one after the other, we can find a $(k - 2)$-degenerate order of all the vertices of B from

the variable gadgets. Then we precede this with the vertices of B in clause gadgets in any order.

To check that this is a $(k-2)$-degenerate order, we need only consider the vertices in the clause gadgets and their neighbours. For each vertex of B in a clause gadget, the only vertices before it in the order are its neighbours that are also in clause gadgets and in B (and there are at most $k-2$ of these, since only one vertex from each clause gadget is in A). The only vertices in a variable gadget with neighbours in a clause gadget are in a top set. However, a vertex in a top set belongs to B if and only if we have a true decomposition of the variable gadget, in which case its neighbour in the clause gadget belongs to A. We conclude that (A, B) is a k-degenerate decomposition of G.

Now suppose that G has a k-degenerate decomposition (A, B). We know, by Lemma 6, that the restriction of (A, B) to a variable gadget is either a true or false decomposition. We assign the value true or false to x_i according to the decomposition of the variable gadget G_i. We know that exactly one vertex x_i^j in each clause gadget F_j is in A (otherwise B contains a clique on k vertices or A is not independent). If we can prove that x_i^j is in A if and only if G_i has a true decomposition, then we have assigned exactly one variable from each clause the value true.

If x_i^j is in A, then G_i must have a true decomposition, otherwise the vertices in the top set adjacent to x_i^j are also in A. If x_i^j is in B, then G_i must have a false decomposition, otherwise the $k-1$ vertices in the top set adjacent to x_i^j are in B and so are $k-2$ vertices in the adjacent odd-level clique, and all these vertices together induce a graph in which every vertex has degree at least $k-1$, which cannot be a subgraph of a $(k-2)$-degenerate graph. We conclude that our truth assignment is good. □

5 | AN APPLICATION: RECONFIGURATIONS OF VERTEX COLOURINGS

Our interest in finding k-degenerate decompositions stems from an open problem in the area of graph reconfigurations. For a graph G and an integer $k \geq 1$, the k-colouring reconfiguration graph $R_k(G)$ has vertex set consisting of all possible k-colourings of G and two vertices of $R_k(G)$ are adjacent if and only if the corresponding k-colourings differ on exactly one vertex. The following problem has been the subject of much study; see e.g. [6-8,19,24,31]:

Given a graph G on n vertices and two k-colourings α and β of G, find a path (if one exists) in $R_k(G)$ between α and β.

We will consider G to be connected since finding a path from α to β in a disconnected graph can be divided into the problem of finding paths between the restrictions of α and β to each component. In this section, we are concerned with determining, for every pair (k, Δ), the complexity of this problem on graphs with maximum degree Δ. Later in this section, we will prove the following result using Theorem 7.

Proposition 1. *Let G be a connected graph on n vertices with maximum degree $\Delta \geq 3$. Then it is possible to find a path in $R_{\Delta+1}(G)$ (if one exists) between any two given $(\Delta+1)$-colourings in $O(n^2)$ time.*

In [24, Theorem 6], three of the authors of the current article proved Proposition 1 in all cases except where the input graph G is Δ-regular. Their argument used the fact that if G has maximum degree Δ, but is not Δ-regular, then G is $(\Delta - 1)$-degenerate. In this case it is possible to translate a structural result of Mihók [41] (also proved by Wood [47]) into an $O(n^2)$-time algorithm, as shown in [24]. However, this does not work if G is Δ-regular.

The following theorem and proof demonstrate that Proposition 1 indeed fills the gap left by past work on this problem.

Theorem 9. *Let $\Delta \geq 0$ be a (fixed) integer. Let G be a connected graph on n vertices with maximum degree Δ. The problem of finding a path (if one exists) between two k-colourings α and β in $R_k(G)$ is*

- *$O(n)$-time solvable if $1 \leq k \leq 3$;*
- *$O(n^2)$-time solvable if $k \geq 4$ and $0 \leq \Delta \leq k - 1$;*
- PSPACE-*hard if $k \geq 4$ and $\Delta \geq k$.*

Proof. In [31], the problem was shown to be solvable in $O(n + m)$ time on (general) graphs with n vertices and m edges for $k \leq 3$. An $O(n^2)$ time algorithm for the case where $k \geq 4, 0 \leq \Delta \leq k - 2$ was presented in [19]. In [7], PSPACE-hardness for $k \geq 4, \Delta \geq k$ was proved. This leaves only the case where $k \geq 4$ and $\Delta = k - 1$, which follows from Proposition 1. □

It was already known [24, Theorem 2] that for every connected graph on n vertices with maximum degree $\Delta \geq 3$, there is a path of length $O(n^2)$ between any two given $(\Delta + 1)$-colourings in $R_{\Delta+1}(G)$ unless one or both of the $(\Delta + 1)$-colourings is an isolated vertex in $R_{\Delta+1}(G)$. To prove Proposition 1, we have to show how to *find* such paths between colourings in $R_{\Delta+1}(G)$ in $O(n^2)$ time. Apart from using Theorem 7, this requires us to replace several structural lemmas of [24] by their algorithmic counterparts. As we have also managed to simplify some of the arguments from [24], we present a self-contained proof of Proposition 1. In fact, most of the work will be done in the proofs of a series of lemmas.

At several places in these proofs we seek to show that from some given colouring α of a graph G, we can find a path in $R_{\Delta+1}(G)$ to another colouring with some specified property. Rather than explicitly referring to paths in $R_{\Delta+1}(G)$, we think, equivalently, in terms of *recolouring* vertices of G one by one to turn α into the colouring we require.

We now define a number of terms that we will use to describe G and its vertices with respect to a $(\Delta + 1)$-colouring α. A vertex v is *locked* by α if Δ distinct colours appear on its neighbours; note that in this case every neighbour of v has a unique colour. A vertex that is not locked is *free*. Clearly a vertex can be recoloured only if it is free. A vertex v is *superfree* if there is a colour $c \neq \Delta + 1$ such that neither v nor any of its neighbours is coloured c, that is, $c \neq \alpha(u)$ for any u in the closed neighbourhood $N_G[v] = \{u | uv \in E\} \cup \{v\}$ of v. A vertex v can be recoloured with a colour other than $\Delta + 1$ if and only if v is superfree. For any two distinct colours $j, k \in \{1, ..., \Delta + 1\}$, a (j, k)-*component* is a connected component in the subgraph of G induced by the vertices coloured j or k. As we continually recolour, these terms should be assumed to be used with respect to the current colouring unless specified otherwise. We let L_α denote the set of vertices u with colour $\alpha(u) = \Delta + 1$. We say that α has the *lock-property* if L_α is nonempty, and every $u \in L_\alpha$ is locked and so are all its neighbours.

If α has the lock-property and u_i is a vertex in L_α, then, for every $j, k \in \{1, ..., \Delta\}, j \neq k$, we denote the unique neighbour of u_i coloured j by $v_{i,j}$, and the graph $[\alpha]G_i^{j,k}$ is the (j,k)-component containing $v_{i,j}$. We note that $[\alpha]G_i^{j,k}$ and $[\alpha]G_i^{k,j}$ may or may not be equal. We will just write $G_i^{j,k}$ if there is no ambiguity. We say that we *compact*α if we determine a path in $R_{\Delta+1}(G)$ from α to a $(\Delta+1)$-colouring α^* of G with $|L_{\alpha^*}| < |L_\alpha|$.

The crucial ideas in the proof of Proposition 1 are that, first, for a $(\Delta+1)$-colouring α of G that is not an isolated vertex in $R_{\Delta+1}(G)$, it is possible to compact α in $O(n)$ time and so, by repetition, obtain a colouring that only uses Δ colours, and, second, that finding paths between pairs of colourings in $R_{\Delta+1}(G)$ that each use only Δ colours is straightforward.

Lemma 7. *Let G be a connected graph on n vertices with maximum degree $\Delta \geq 3$. Let α be a $(\Delta+1)$-colouring of G such that $L_\alpha \neq \emptyset$. Given a free vertex v in $N[L_\alpha]$, we can compact α in $O(1)$ time.*

Proof. Let $u \in L_\alpha$ be a neighbour of v. First suppose that u is free. Let $c \neq \Delta+1$ be a colour not used on $N_G[u]$. We can determine c in $O(1)$ time, as Δ is a constant. Then we can recolour u with colour c. Now assume that u is not free, that is, u is locked. We determine a colour c' not used on $N_G[v]$ in $O(1)$ time and note that $c' \neq \Delta+1$. We recolour v with colour c', and now u is free and can be recoloured with $\alpha(v) \neq \Delta+1$. Hence, we have compacted α in $O(1)$ time. □

Lemma 8. *Let G be a connected graph on n vertices with maximum degree $\Delta \geq 3$. Let α be a $(\Delta+1)$-colouring of G. Let j and k be distinct colours in $\{1, ..., \Delta\}$. Given a (j,k)-component D such that no vertex coloured j in D has a neighbour in L_α, we can recolour G in $O(|V(D)|)$ time from α to the $(\Delta+1)$-colouring α' with*

- *$\alpha'(v) = \alpha(v)$ for all $v \notin V(D)$,*
- *$\alpha'(v) = j + k - \alpha(v)$ for all $v \in V(D)$ (i.e. the colours on D are* swapped*).*

Proof. We recolour all vertices of D that have colour j with colour $\Delta+1$. This yields a new $(\Delta+1)$-colouring, as none of these vertices has a neighbour in L_α. We then recolour all vertices of D that have colour k with the colour j. Again this is a valid operation as, by the choice of D, all their neighbours that were given colour j by α are now coloured $\Delta+1$. We finally recolour all vertices of D that were given colour j by α (and so have $\Delta+1$ in the current colouring) with the colour k. This yields the $(\Delta+1)$-colouring α'. Moreover, the running time of doing this is linear in the size of D. □

Lemma 9. *Let G be a connected graph on n vertices with maximum degree $\Delta \geq 3$. Let α be a $(\Delta+1)$-colouring of G that has the lock-property. Let u_i be a vertex in L_α and let j and k be two distinct colours in $\{1, ..., \Delta\}$. If $G_i^{j,k}$ is* not *a path where each end-vertex is locked and no vertex is superfree, then we can compact α in $O(n)$ time.*

Proof. We start with two observations. If a vertex $v \in V(G_i^{j,k})$ has degree at least 2 in $G_i^{j,k}$, then v has two neighbours with the same colour, so it is free, and therefore, by the lock-property, not adjacent to L_α. If a vertex $v \in V(G_i^{j,k})$ has degree at least 3 in $G_i^{j,k}$, then v is, in fact, superfree.

As $v_{i,j}$ is in $N_G[u_i]$, by the lock-property, we know that $v_{i,j}$ is locked. Hence $v_{i,j}$ has degree 1 in $G_i^{j,k}$. We now perform a breadth-first search in $G_i^{j,k}$ starting from $v_{i,j}$. This takes $O(n)$ time, as Δ is a constant. We stop if we find a superfree vertex w or else if we have visited all vertices of $G_i^{j,k}$.

First suppose that we found a superfree vertex w. As all vertices closer to $v_{i,j}$ than w in $G_i^{j,k}$ are not superfree, they must have degree 2 and form a path P from $v_{i,j}$ to w. As the internal vertices of P have degree 2 in $G_i^{j,k}$, they are free and so have no neighbour in L_α by the lock-property. Since w is superfree, there is some colour $c \neq \Delta + 1$ with which we can recolour w. As colours j and k appear on $N_G[w]$, we find that $c \notin \{j, k\}$. After we recolour w with c, we note that $G_i^{j,k}$ (defined with respect to this new colouring) is a (j, k)-component where no vertex coloured k has a neighbour in L_α. We apply Lemma 8 and note that now u_i has no neighbour coloured j. We therefore recolour u_i with j and have compacted α in $O(n)$ time.

Now suppose that we found that no vertex in $G_i^{j,k}$ is superfree. Then no vertex of $G_i^{j,k}$ has degree more than 2. Hence $G_i^{j,k}$ is a path or cycle. Since $v_{i,j}$ has degree 1, we find that $G_i^{j,k}$ must be a path. Let z be the end-vertex of $G_i^{j,k}$ other than $v_{i,j}$. As no vertex of the path $G_i^{j,k}$ is superfree and $v_{i,j}$ is locked, z must be free by the assumption of the lemma. Hence, every vertex of $G_i^{j,k}$ apart from $v_{i,j}$ is free, which means that no vertex of $G_i^{j,k}$ with colour k has a neighbour in L_α by the lock-property. We apply Lemma 8. Afterwards we can recolour u_i with colour j to compact α in $O(n)$ time. □

Lemma 10. *Let G be a connected graph on n vertices with maximum degree $\Delta \geq 3$. Let α be a $(\Delta + 1)$-colouring of G that has the lock-property. Let u_{i_1} and u_{i_2} be two vertices in L_α (possibly $u_{i_1} = u_{i_2}$) and let j_1, j_2, k_1 and k_2 be in $\{1, ..., \Delta\}, j_1 \neq k_1, j_2 \neq k_2$. If $G_{i_1}^{j_1,k_1}$ and $G_{i_2}^{j_2,k_2}$ are two distinct paths, each with locked end-vertices and no superfree vertices, then $G_{i_1}^{j_1,k_1}$ and $G_{i_2}^{j_2,k_2}$ do not intersect on a free vertex.*

Proof. For contradiction, suppose that $G_{i_1}^{j_1,k_1}$ and $G_{i_2}^{j_2,k_2}$ intersect on a free vertex w. Then w is an internal vertex on each of $G_{i_1}^{j_1,k_1}$ and $G_{i_2}^{j_2,k_2}$. Moreover, the two neighbours of w in $G_{i_1}^{j_1,k_1}$ are disjoint from the two neighbours of w in $G_{i_2}^{j_2,k_2}$, otherwise $\{j_1, k_1\} = \{j_2, k_2\}$, in which case $G_{i_1}^{j_1,k_1} = G_{i_2}^{j_2,k_2}$, a contradiction. Thus w has four neighbours that use only two colours between them. So w has at most $\Delta - 2$ colours in its neighbourhood. This means that w is superfree, a contradiction. □

Lemma 11. *Let G be a connected graph on n vertices with maximum degree $\Delta \geq 3$. Let α be a $(\Delta + 1)$-colouring of G that has the lock-property. Let u_i be a vertex in L_α such that for every two distinct colours h, h' in $\{1, ..., \Delta\}$, $G_i^{h,h'}$ and $G_i^{h',h}$ are paths that have locked end-vertices and no superfree vertices. Let $j, k \in \{1, ..., \Delta\}$ be such that $G_i^{j,k}$ and $G_i^{k,j}$ are distinct paths, and, moreover, $G_i^{j,k}$ does not contain exactly two vertices. Then it is possible to compact α in $O(n)$ time.*

Proof. Since $v_{i,j} \in N[u_i]$ is locked by the lock-property, $v_{i,j}$ has exactly one neighbour s coloured k. This means that $G_i^{j,k}$ has at least two vertices, and thus at least three vertices by the assumption of the lemma. Hence s has another neighbour with colour j. This means that s is free. Note that s is not superfree by the assumption of the claim.

As $\Delta \geq 3$, there exists a colour $c \notin \{j, k, \Delta + 1\}$ and, as s is not superfree, s has a neighbour t coloured c. First suppose that t is locked. Then, by definition, t has a

neighbour $u_g \in L_\alpha$ with colour $\Delta + 1$. If $u_g = u_i$, then $G_i^{j,k}$ and $G_i^{c,k}$ intersect on the free vertex s, contradicting Lemma 10. Hence, $u_g \neq u_i$. Then we recolour s with $\Delta + 1$. This is possible, as s is free and any vertices adjacent to vertices in L_α are locked by the lock-property. We then recolour $v_{i,j}$ with colour k. As t is locked by α, we find that α has coloured exactly one neighbour of t with colour j and exactly one neighbour (namely s) with colour k. If t is adjacent to $v_{i,j}$ then this is the neighbour of t that is coloured j by α; in this case we recolour t with colour j. If t is nonadjacent to $v_{i,j}$ then we recolour t with colour k. Thus we can recolour t either with colour j or with colour k. Now both u_g and u_i are free and can be recoloured. In this way in $O(n)$ time we have reduced the total number of vertices coloured $\Delta + 1$ by $2 - 1 = 1$, that is, we have compacted α.

Now suppose that t is free. As the end-vertex of $G_i^{j,k}$ other than $v_{i,j}$ is locked by assumption, it has a neighbour $u_\ell \in L_\alpha$. If $u_\ell = u_i$, then $G_i^{j,k} = G_i^{k,j}$, which is not possible by assumption. Hence we have $u_\ell \neq u_i$. We now recolour t with colour $\Delta + 1$. This is possible, as t is free and thus has no neighbour in L_α by the lock-property. As s is not superfree, α colours exactly two neighbours of s with the same colour, which is j. Hence, t is the only neighbour of s which α colours with colour c. Hence, after recolouring t with colour $\Delta + 1$, we can recolour s with c. Thus we can recolour $v_{i,j}$ with k. As all the internal vertices in $G_i^{j,k}$ were free, by the lock-property, they had no neighbours in L_α. We now apply Lemma 8 (with the roles of j and k reversed) to the subpath of $G_i^{j,k} \setminus \{s\}$ containing a neighbour of u_ℓ. Now both u_i and u_ℓ can be recoloured as they have no neighbour coloured j. In this way, in $O(n)$ time, we have reduced the number of vertices coloured $\Delta + 1$ by $2 - 1 = 1$, that is, we have compacted α. □

Lemma 12. *Let G be a connected graph on n vertices with maximum degree $\Delta \geq 3$ that is not isomorphic to $K_{\Delta+1}$. Let α be a $(\Delta + 1)$-colouring of G that is not an isolated vertex in $R_{\Delta+1}(G)$ such that $L_\alpha \neq \emptyset$. Then it is possible to compact α in $O(n)$ time.*

Proof. We note that G has $O(n)$ edges as Δ is a fixed constant. Let $L_\alpha = \{u_1, ..., u_p\}$ for some $p \geq 1$.

Our algorithm. If α does not have the lock-property then we are done by Lemma 7. We may therefore assume that α has the lock-property. As α is not an isolated vertex in $R_{\Delta+1}(G)$, G has at least one free vertex x. Let P be a shortest path in G from x to a vertex in L_α, say to u_1. We may assume that x is chosen such that x is the free vertex on P closest to u_1. Then every internal vertex of P is locked and coloured with a colour in $\{1, ..., \Delta\}$. Moreover, the only vertex of P with a neighbour in L_α is the neighbour of u_1. By definition, any locked vertex not in L_α has a neighbour with colour $\Delta + 1$, so it is adjacent to a vertex in L_α. Hence P has at most two edges. As x is free, but by the lock-property every vertex in $N_G[u_1]$ is locked, it follows that u_1 and x are not adjacent. Therefore, P contains exactly two edges. Without loss of generality, we may assume that the middle vertex is $v_{1,1}$, which has colour 1 by definition, and that x is coloured 2. In particular note that in this case $x \in V(G_1^{1,2})$.

We may assume, for any two distinct colours h, h' in $\{1, ..., \Delta\}$, that both $G_1^{h,h'}$ and $G_1^{h',h}$ are paths whose end-vertices are locked and that do not contain any superfree vertices, otherwise we apply Lemma 9 and are done. As $G_1^{1,2}$ is a path whose end-vertices are locked and $x \in V(G_1^{1,2})$ is free, $G_1^{1,2}$ has at least three vertices. Hence, we may assume that $G_1^{1,2} = G_1^{2,1}$, otherwise we may apply Lemma 11 and are done.

Let $H^{2,3}$ be the (2, 3)-component of G containing x. As x is not superfree and has two neighbours coloured 1, x has only one neighbour coloured 3. Hence x has degree 1 in $H^{2,3}$. If $H^{2,3}$ is not a path, then let w be the vertex with degree at least 3 in $H^{2,3}$ that is closest to x. As w has three neighbours coloured alike, w is superfree. This means we can recolour w with a colour $c \neq \Delta + 1$. Note that $c \notin \{2, 3\}$.

This recolouring of w may have altered the graph $G_1^{1,2}$. Note that w is not part of $G_1^{1,2}$ before it is recoloured since, unlike vertices on $G_1^{1,2}$, it is superfree. It becomes part of $G_1^{1,2}$ after recolouring only if $c = 1$ (since we know $c \neq 2$). If this occurs, then $G_1^{1,2}$ is no longer a path, and we apply Lemma 9 and are done in $O(n)$ time. Hence suppose that $G_1^{1,2}$ is (still) a path and note that the recolouring of w also changed $H^{2,3}$ into a path (if $H^{2,3}$ was not already a path). For simplicity, we still call the current colouring α.

The internal vertices of the path $H^{2,3}$ each have two neighbours coloured alike. Hence, they are not locked and therefore all but at most one vertex of $H^{2,3}$ is free. By the lock-property, every neighbour of a vertex in L_α is locked. Consequently, no internal vertex of $H^{2,3}$ has a neighbour in L_α. Let x' be the end-vertex of $H^{2,3}$ other than x. Then $\alpha(x')$ is either 2 or 3. If $\alpha(x') = 2$, then we apply Lemma 8 with $j = 3$ and $k = 2$. If $\alpha(x') = 3$, then we apply Lemma 8 with $j = 2$ and $k = 3$ (as x is free and thus has no neighbour in L_α either). Let β be the resulting colouring.

We now proceed by applying a similar procedure to β to the one we applied to α. If β does not have the lock-property, then we are done by Lemma 7, so we may assume that β does have the lock-property. Recall that $v_{1,2}$ and $v_{1,3}$ are locked by α. Since at most one vertex in $H^{2,3}$ is locked by α, it follows that at most one vertex in $\{v_{1,2}, v_{1,3}\}$ is in $H^{2,3}$. If $v_{1,2}$ is in $H^{2,3}$, but $v_{1,3}$ is not then $\beta(v_{1,2}) = \beta(v_{1,3}) = 3$, so u_1 is not locked by β, contradicting the lock-property. Therefore $v_{1,2}$ is not in $H^{2,3}$, and similarly $v_{1,3}$ is not in $H^{2,3}$, so $\beta(v_{1,2}) = 2$ and $\beta(v_{1,3}) = 3$. We may assume that $[\beta]G_1^{1,2}$ and $[\beta]G_1^{2,1}$ are paths whose end-vertices are locked and that do not contain any superfree vertices, otherwise we apply Lemma 9 and are done. Note that $[\alpha]G_1^{1,2} = [\alpha]G_1^{2,1}$ consists of a path which contains the vertices $v_{1,1}$, x, and $v_{1,2}$ in that order and α colours these vertices 1, 2 and 2, respectively. Therefore $v_{1,2}$ must have a neighbour v' in $[\alpha]G_1^{2,1}$ that α colours with colour 1, and this must be an internal vertex of $[\alpha]G_1^{2,1}$, so by the lock-property, it cannot be adjacent to a vertex in L_α. Since $L_\alpha = L_\beta$ it follows that v' has no neighbours in L_β, and so it must be free in β. As $[\beta]G_1^{2,1}$ is a path whose end-vertices are locked and $v_{1,2} \in V([\beta]G_1^{2,1})$ is free, $[\beta]G_1^{2,1}$ has at least three vertices. Thus we may assume that $[\beta]G_1^{1,2} = [\beta]G_1^{2,1}$, otherwise we apply Lemma 11 and are done.

As $v_{1,1}$ is locked by β, we find that $v_{1,1}$ has a neighbour z with $\beta(z) = 2$. Hence, z belongs to $[\beta]G_1^{1,2}$. Recall that $\alpha(x) = 2$. As $v_{1,1}$ is locked by α, we find that $v_{1,1}$ has no other neighbour that got colour 2 in the colouring α, by the lock-property. Note that $\beta(x) = 3$. Hence, in order for $v_{1,1}$ to have a neighbour coloured 2 by β, it must be the case that z belongs to $H^{2,3}$ and thus $\alpha(z) = 3$. As $v_{1,2}$ is not in $H^{2,3}$, we find that $z \neq v_{1,2}$. Then z is an internal vertex of $[\beta]G_1^{1,2} = [\beta]G_1^{2,1}$. Thus z has two neighbours coloured 1. If z is an internal vertex of $H^{2,3}$, then β colours two other neighbours of z with colour 3. This implies that z is superfree, contradicting the fact that $[\beta]G_1^{1,2}$ has no superfree vertices. Thus z is the end-vertex of $H^{2,3}$ other than x, that is, $z = x'$.

We now let y be the first vertex of $[\beta]G_1^{1,2}$ that is not in $[\alpha]G_1^{1,2}$ when traversing $[\beta]G_1^{1,2}$ from $v_{1,2}$. Note that such a vertex y exists, as z belongs to $[\beta]G_1^{1,2}$ but not to $[\alpha]G_1^{1,2}$ (as $\alpha(z) = 3$). Thus $\alpha(y) \neq \beta(y)$, so y belongs to $H^{2,3}$. The end-vertices of $[\beta]G_1^{1,2}$ are $v_{1,1}$ and $v_{1,2}$, neither of which belong to $H^{2,3}$. Hence, y is an inner vertex of $[\beta]G_1^{1,2}$. If y is an

internal vertex of both $H^{2,3}$ and $[\beta]G_1^{1,2}$, then y would be superfree, contradicting the fact that $[\beta]G_1^{1,2}$ has no superfree vertices. This means that y is an end-vertex of $H^{2,3}$. As $\beta(x) = 3$ while $\beta(y) = 1$ or $\beta(y) = 2$, we find that $y \neq x$. It follows that $y = z = x'$, and so $\beta(y) = 2$.

Consider the vertex z' on $[\beta]G_1^{1,2}$ reached immediately before $y = z = x'$ when traversing $[\beta]G_1^{1,2}$ from $v_{1,2}$. Then $\alpha(z') = \beta(z') = 1$. As $\alpha(v_{1,2}) = \alpha(y) = 2$, we find that z' is an inner vertex of $[\alpha]G_1^{1,2}$. It follows that z' is a free vertex. Note that z is adjacent to $v_{1,1}$ and z' and that $\alpha(z) = 3$ and $\alpha(v_{1,1}) = \alpha(z') = 1$. Therefore z' is a vertex of $[\alpha]G_1^{1,3}$ and so $[\alpha]G_1^{1,3}$ and $[\alpha]G_1^{1,2}$ intersect on the free vertex z'. By Lemma 10, it follows that $[\alpha]G_1^{1,3}$ is not a path whose end-vertices are locked and that does not contain any superfree vertices, and so we are done by applying Lemma 9. This completes the description of the algorithm.

The correctness of our algorithm follows directly from its description. Hence it remains to discuss its runtime.

Runtime analysis. We first compute the set L_α in $O(n)$ time, as Δ is a constant. We then apply Lemma 7 on each u_i. As this takes $O(1)$ time per vertex, obtaining the lock-property takes $O(n)$ in total. As Δ is a constant, for a given pair (i, j), the vertex $v_{i,j}$ can be found in $O(1)$ time. Moreover, for a given triple (i, j, k), we can compute $G_i^{j,k}$ in $O(n)$ time. As Δ is a constant, we can find a free vertex x in $O(n)$ time.

We can find the path P to a vertex in L_α, which we assumed was u_1, by using a breadth-first search starting from x. As Δ is a constant, this takes $O(n)$ time. We may also assume that x is the free vertex on P closest to u_1 on this path, as otherwise we can replace x by some other free vertex of P in $O(n)$ time.

We check in $O(n)$ time whether $G_1^{1,2}$ is a path whose end-vertices are locked and that does not contain any superfree vertices. We check the same in $O(n)$ time for $G_1^{2,1}$. If we find that for at least one of these graphs this is not the case, then our algorithm applies Lemma 9 (either with $i = j = 1, k = 2$ or with $i = k = 1, j = 2$), and we are done in $O(n)$ time. So suppose this is the case for both $G_1^{1,2}$ and $G_1^{2,1}$. Then we check in $O(n)$ time whether $G_1^{1,2} = G_1^{2,1}$. If not, then our algorithm applies Lemma 11 (with $i = j = 1, k = 2$), and we are done in $O(n)$ time. So suppose that $G_1^{1,2} = G_1^{2,1}$.

Recolouring the vertex w in $H^{2,3}$ (if it exists) and applying Lemma 8 (either with $j = 3, k = 2$ or with $j = 2, k = 3$) takes $O(n)$ time. So far we have used $O(n)$ time. Hence, it takes $O(n)$ time to do the similar procedure for the colouring β obtained after applying Lemma 8. We find the vertices z and z' in $O(1)$ time. Afterwards we apply Lemma 9, which takes $O(n)$ time. So we used $O(n)$ time in total. This completes the proof of the lemma. □

We are now ready to prove Proposition 1 for graphs G with maximum degree $\Delta \geq 3$ by following the arguments from [24] without the requirement that G is $(\Delta - 1)$-degenerate (i.e. we allow G to be Δ-regular). So the proof is similar to the proof in [24] for $(\Delta - 1)$-degenerate graphs except that we use Theorem 7 instead of its algorithmic counterpart for $(\Delta - 1)$-degenerate graphs. To show this we give a self-contained proof.

Proposition 1 (Restated). *Let G be a connected graph on n vertices with maximum degree $\Delta \geq 3$. Then it is possible to find a path in $R_{\Delta+1}(G)$ (if one exists) between any two given $(\Delta + 1)$-colourings in $O(n^2)$ time.*

Proof. We use induction on Δ, and, to begin, use the fact that the result is known for $\Delta = 2$ (3-colourings of paths and cycles, see [31] and note that, as discussed above in the proof of Theorem 9, their result was more general than this). For $\Delta \geq 3$, assume that we have an $O(n^2)$ time algorithm for connected graphs on n vertices with maximum degree $\Delta - 1$. Let α and β be two $(\Delta + 1)$-colourings of G. We can check in $O(n)$ time whether α or β is an isolated vertex in $R_{\Delta+1}(G)$ and, clearly, if one or other is, then no path between them exists. Otherwise we apply Lemma 12 $O(n)$ times, and in $O(n^2)$ time we can find a path from α to some Δ-colouring γ_1 and a path from β to some Δ-colouring γ_2. By Theorem 7 we can find in $O(n^2)$ time a partition $\{S_1, S_2\}$ of $V(G)$ such that S_1 is an independent set and S_2 induces a $(\Delta - 2)$-degenerate graph, which we denote by H. We modify the pair (S_1, S_2) in $O(n^2)$ time by moving vertices from S_2 to S_1 until a superset of S_1 that is a maximal independent set is obtained. We denote the modified pair $(S_1^{'}, S_2^{'})$

Let γ_1^H and γ_2^H be the Δ-colourings of H that are the restrictions of γ_1 and γ_2, respectively, to $S_2^{'}$. Let $\gamma_1^{'}$ and $\gamma_2^{'}$ be the $(\Delta + 1)$-colourings obtained from γ_1 and γ_2, respectively, by recolouring every vertex in $S_1^{'}$ with the colour $\Delta + 1$. As $S_1^{'}$ is maximal, H has maximum degree at most $\Delta - 1$. We apply the induction hypothesis to find in $O(n^2)$ time a path between the two Δ-colourings γ_1^H and γ_2^H in $R_\Delta(H)$ (note that neither is an isolated vertex of $R_\Delta(H)$ since H is $(\Delta - 2)$-degenerate). Note that this immediately translates into a path between $\gamma_1^{'}$ and $\gamma_2^{'}$ in $R_{\Delta+1}(G)$. Hence we obtain a path between α and β in $R_{\Delta+1}(G)$. This completes the proof. □

6 | FUTURE WORK

In this section we pose two open problems. We have proven that for every integer $k \geq 3$, the problem of finding a k-degenerate decomposition is polynomial-time solvable on graphs of maximum degree k and NP-hard for graphs of maximum degree $2k - 2$ (by generalizing the hardness proof of [49] for $k = 3$). This brings us to our first open problem.

Open Problem 1. *Determine, for every integer $k \geq 4$, the complexity of finding a k-degenerate decomposition for graphs of maximum degree $k + 1$.*

Our second open problem is related to Theorem 3. Recall that this theorem states that for every three integers $k \geq 3$ and $p, q \geq 0$ with $p + q = k - 2$, the vertex set of every connected graph of maximum degree k that is not isomorphic to K_{k+1} can be partitioned into two sets A and B, where A induces a p-degenerate subgraph of maximum size and B induces a q-degenerate subgraph. Our algorithms in Sections 2 and 3 form an algorithmic version of Theorem 5, which is a special case of Theorem 3 in which $p = 0$ and $q = k - 2$.

Open Problem 2. *Does there exist an algorithmic version of Theorem 3 similar to our algorithmic version of Theorem 5?*

ACKNOWLEDGEMENTS

We thank a reviewer for all the helpful comments for improving our article. An extended abstract of this article appeared in the proceedings of MFCS 2017 [3].

ORCID

Marthe Bonamy http://orcid.org/0000-0001-7905-8018
Konrad K. Dabrowski https://orcid.org/0000-0001-9515-6945
Carl Feghali http://orcid.org/0000-0001-6727-7213
Matthew Johnson http://orcid.org/0000-0002-7295-2663
Daniël Paulusma https://orcid.org/0000-0001-5945-9287

REFERENCES

1. A. Agrawal, S. Gupta, S. Saurabh, and R. Sharma, *Improved algorithms and combinatorial bounds for independent feedback vertex set*, Proc. IPEC 2016, LIPIcs. **63** (2017), 2:1–2:14.
2. B. Baetz and D. R. Wood, *Brooks' vertex-colouring theorem in linear time*, CoRR. abs/1401.8023, 2014.
3. M. Bonamy, K. K. Dabrowski, C. Feghali, M. Johnson, and D. Paulusma, *Recognizing graphs close to bipartite graphs*, Proc. MFCS 2017, LIPIcs. **83** (2017), 70:1–70:14.
4. M. Bonamy, K. K. Dabrowski, C. Feghali, M. Johnson, and D. Paulusma, *Independent feedback vertex sets for graphs of bounded diameter*, Inf. Process. Lett. **131** (2018), 26–32.
5. M. Bonamy, K. K. Dabrowski, C. Feghali, M. Johnson, and D. Paulusma, *Independent feedback vertex set for P_5 -free graphs*, Algorithmica **81** (2019), 1342–1369.
6. M. Bonamy, M. Johnson, I. Lignos, V. Patel, and D. Paulusma. *Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs*, J. Combin. Optim. **27** (2014), no. 1, 132–143.
7. P. S. Bonsma and L. Cereceda, *Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances*, Theor. Comput. Sci. **410** (2009), no. 50, 5215–5226.
8. P. S. Bonsma, A. E. Mouawad, N. Nishimura, and V. Raman, *The complexity of bounded length graph recoloring and CSP reconfiguration*, Proc. IPEC 2014, LNCS. **8894** (2014), 110–121.
9. O. V. Borodin, *On decomposition of graphs into degenerate subgraphs*, Diskretnyĭ Analiz. **28** (1976), 3–11. (Russian).
10. O. V. Borodin and A. N. Glebov, *On the partition of a planar graph of girth 5 into an empty and an acyclic subgraph*, Diskretnyĭ Analiz i Issledovanie Operatsiĭ. Seriya 1. **8** (2001), 34–53 (in Russian).
11. O. V. Borodin, A. V. Kostochka, and B. Toft, *Variable degeneracy: extensions of Brooks' and Gallai's theorems*, Discrete Math. **214** (2000), no. 1–3, 101–112.
12. A. Brandstädt, S. Brito, S. Klein, L. T. Nogueira, and F. Protti, *Cycle transversals in perfect graphs and cographs*, Theor. Comput. Sci. **469** (2013), 15–23.
13. A. Brandstädt, P. L. Hammer, V. B. Le, and V. V. Lozin, *Bisplit graphs*, Discrete Math. **299** (2005), no. 1–3, 11–32.
14. A. Brandstädt, V. B. Le, and T. Szymczak, *The complexity of some problems related to graph 3-colorability*, Discrete Appl. Math. **89** (1998), no. 1–3, 59–73.
15. R. L. Brooks, *On colouring the nodes of a network*, Math. Proc. Camb. Philos. Soc. **37** (1941), no. 2, 194–197.
16. L. Cai and D. G. Corneil, *A generalization of perfect graphs - i -perfect graphs*, J. Graph Theory **23** (1996), no. 1, 87–103.
17. P. A. Catlin, *Brooks' graph-coloring theorem and the independence number*, J. Combin. Theory Ser. B **27** (1979), no. 1, 42–48.
18. P. A. Catlin and H. -J. Lai, *Vertex arboricity and maximum degree*, Discrete Math. **141** (1995), no. 1–3, 37–46.
19. L. Cereceda, *Mixing graph colourings*. Ph.D. Thesis, London School of Economics, 2007.
20. G. Chartrand and H. V. Kronk, *The point-arboricity of planar graphs*, J. Lon. Math. Soc. **s1-44** (1969), no. 1, 612–616.
21. K. K. Dabrowski, V. V. Lozin, and J. Stacho, *Stable-Π partitions of graphs*, Discrete Appl. Math. **182** (2015), 104–114.
22. F. Dross, M. Montassier, and A. Pinlou, *Partitioning a triangle-free planar graph into a forest and a forest of bounded degree*, Eur. J. Combin. **66** (2017), 81–94.
23. T. Feder, P. Hell, S. Klein, and R. Motwani, *List partitions*, SIAM J. Discrete Math. **16** (2003), no. 3, 449–478.
24. C. Feghali, M. Johnson, and D. Paulusma, *A reconfigurations analogue of Brooks' Theorem and its consequences*, J. Graph Theory **83** (2016), no. 4, 340–358.

25. M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman & Co., New York, NY, 1979.
26. M. R. Garey, D. S. Johnson, and L. J. Stockmeyer, *Some simplified NP-complete graph problems*, Theor. Comput. Sci. **1** (1976), no. 3, 237–267.
27. D. L. Grinstead, P. J. Slater, N. A. Sherwani, and N. D. Holmes, *Efficient edge domination problems in graphs*, Inf. Process. Lett. **48** (1993), no. 5, 221–228.
28. M. Grötschel, L. Lovász, and A. Schrijver, *Polynomial algorithms for perfect graphs*, Ann. Discrete Math. **21** (1984), 325–356.
29. S. L. Hakimi, E. F. Schmeichel, and J. Weinstein, *Partitioning planar graphs into independent sets and forests*, Congr. Numer. **78** (1990), 109–118.
30. C. T. Hoàng and V. B. Le, *On P_4-transversals of perfect graphs*, Discrete Math. **216** (2000), no. 1–3, 195–210.
31. M. Johnson, D. Kratsch, S. Kratsch, V. Patel, and D. Paulusma, *Finding shortest paths between graph colourings*, Algorithmica. **75** (2016), no. 2, 295–321.
32. K. Kawarabayashi and C. Thomassen, *Decomposing a planar graph of girth 5 into an independent set and a forest*, J. Combin. Theory Ser. B **99** (2009), no. 4, 674–684.
33. A. V. Kostochka, *Upper bounds of chromatic functions of graphs*. Ph.D. Thesis, University of Novosibirsk, 1978 (in Russian).
34. J. Kratochvíl and I. Schiermeyer, *On the computational complexity of $(\mathcal{O}, \mathcal{P})$-partition problems*, Discuss. Math. Graph Theory **17** (1997), no. 2, 253–258.
35. L. Lovász, *A characterization of perfect graphs*, J. Combin. Theory Ser. B **13** (1972), no. 2, 95–98.
36. L. Lovász, *Coverings and coloring of hypergraphs*, Congr. Numer. **VIII** (1973), 3–12.
37. V. V. Lozin, *Between 2- and 3-colorability*, Inf. Process. Lett. **94** (2005), no. 4, 179–182.
38. N. V. R. Mahadev, and U. N. Peled, *Threshold graphs and related topics*, volume 56 of Annals of Discrete Mathematics. North-Holland, Amsterdam, 1995.
39. M. Matamala, *Vertex partitions and maximum degenerate subgraphs*, J. Graph Theory **55** (2007), no. 3, 227–232.
40. C. McDiarmid and N. Yolov, *Recognition of unipolar and generalised split graphs*, Algorithms. **8** (2015), no. 1, 46–59.
41. P. Mihók, *Minimal reducible bounds for the class of k -degenerate graphs*, Discrete Math. **236** (2001), no. 1–3, 273–279.
42. N. Misra, G. Philip, V. Raman, and S. Saurabh, *On parameterized independent feedback vertex set*, Theor. Comput. Sci. **461** (2012), 65–75.
43. P. Ochem, *Negative results on acyclic improper colorings*, Proc. EuroComb 2005 Discr. Math. Theor. Comput. Sci. **AE** (2005), 357–362.
44. Y. Tamura, T. Ito, and X. Zhou, *Algorithms for the independent feedback vertex set problem*, IEICE Trans. Fund. Electr. Commun. Comput. Sci. **E98-A** (2015), no. 6, 1179–1188.
45. C. Thomassen, *Decomposing a planar graph into degenerate graphs*, J. Combin. Theory Ser. B **65** (1995), no. 2, 305–314.
46. C. Thomassen, *Decomposing a planar graph into an independent set and a 3-degenerate graph*, J. Combin. Theory Ser. B **83** (2001), no. 2, 262–271.
47. D. R. Wood, *Acyclic, star and oriented colourings of graph subdivisions*, Discr. Math. Theor. Comput. Sci. **7** (2005), 37–50.
48. Y. Wu, J. Yuan, and Y. Zhao, *Partition a graph into two induced forests*, J. Math. Stud. **29** (1996), 1–6.
49. A. Yang and J. Yuan, *Partition the vertices of a graph into one independent set and one acyclic set*, Discrete Math. **306** (2006), no. 12, 1207–1216.

How to cite this article: M. Bonamy, K. K. Dabrowski, C. Feghali, M. Johnson, and D. Paulusma, *Recognizing graphs close to bipartite graphs with an application to colouring reconfiguration*, J. Graph Theory. 2021;98:81–109. https://doi.org/10.1002/jgt.22683